HyRAL 安全性評価報告書

2011 年 2 月

名古屋大学大学院工学研究科

岩田　哲

要旨

HyRAL は SCIS 2010 において提案されたブロック暗号であり，ブロック長は 128 ビット，鍵長は $128, 129, \ldots, 256$ ビットをサポートしている [15]．128 ビット鍵 HyRAL のラウンド数は 24 であり，$129, 130, \ldots, 256$ ビット鍵の場合は 32 ラウンドである．本報告書は 2011 年 2 月までに得られた HyRAL の安全性評価結果である．評価期間中に得られた主な結果をまとめる．

1. 256 ビット鍵 HyRAL には等価鍵が $2^{51.0}$ 個 ($2^{50.0}$ ペア) 存在することを示した．この等価鍵の存在は全数探索攻撃にかかる計算量が $2^{256} - 2^{50.0}$ であり，2^{256} を下回ることを意味する．したがって 256 ビット鍵 HyRAL は理論的に解読された．

2. 256 ビット鍵 HyRAL の等価鍵の具体例を計算機により導出した．

3. $129, 130, \ldots, 256$ ビット鍵 HyRAL に対する関連暗号攻撃により，128 回のクエリで 256 ビット鍵 HyRAL の秘密鍵の下位 127 ビットを導出できることを示した．

4. $129, 130, \ldots, 256$ ビット鍵 HyRAL を用いて Davies-Meyer 方式により構成した圧縮関数に対し，異なるブロック暗号を用いた圧縮関数同士の衝突を導出できることを示した．

1, 2 の結果は，256 ビット鍵 HyRAL における $2^{51.0}$ 個の弱鍵の存在，256 ビット鍵 HyRAL を用いて Davies-Meyer 方式により構成した圧縮関数に対し，衝突の具体例を導出できること，その圧縮関数を用いて Merkle-Damgård 方式でハッシュ関数を構成した場合，このハッシュ関数に対する衝突の具体例を導出できること，識別成功確率が 2^{-206} である関連鍵識別攻撃が可能であることを意味する．

また，評価期間中に上記以外の問題点を発見していない．

目 次

1 HyRAL の仕様 **1**

1.1 鍵生成アルゴリズム (128 ビット鍵): KGA 4
1.2 鍵生成アルゴリズム (129, 130, . . . , 256 ビット鍵): $\mathrm{KGA}_1, \mathrm{KGA}_2$ . . 4
1.3 データ撹拌関数 (128 ビット鍵): DPA 6
1.4 データ撹拌関数 (129, 130, . . . , 256 ビット鍵): DPA 6
1.5 拡大関数: G_1, G_2 関数 8
1.6 拡大関数: F_1, F_2 関数 9
1.7 基本関数: $f_1, \ldots, f_8$ 関数 11
1.8 鍵割り当てアルゴリズム (128 ビット鍵): KAA 14
1.9 鍵割り当てアルゴリズム (129, 130, . . . , 256 ビット鍵): KAA 16

2 差分攻撃 (128, 192, 256 ビット鍵) **19**

2.1 差分攻撃の概要 19
2.2 HyRAL に対する差分攻撃 20

3 不可能差分攻撃 (128, 192, 256 ビット鍵) **21**

3.1 不可能差分攻撃の概要 21
3.2 HyRAL に対する不可能差分攻撃 21

4 線形攻撃 (128, 192, 256 ビット鍵) **23**

4.1 線形攻撃の概要 23
4.2 HyRAL に対する線形攻撃 24

5 高階差分攻撃 (129, 130, . . . , 256 ビット鍵) **25**

5.1 高階差分攻撃の概要 25
5.2 **129, 130, . . . , 256** ビット鍵 HyRAL に対する高階差分攻撃 26

6 補間攻撃 (128, 192, 256 ビット鍵) **27**

7 代数攻撃 (128, 192, 256 ビット鍵) **28**

8 等価鍵 (256 ビット鍵) **29**

8.1 等価鍵の存在を示すための方針 29
8.2 KGA の差分特性解析 (1) 30
8.3 KGA の差分特性解析 (2) 34
8.4 等価鍵の存在性 36
8.5 等価鍵導出アルゴリズム 37
8.6 アルゴリズムの計算量 43
8.7 計算機による等価鍵の導出 44

8.8 本章のまとめ 49

9 弱鍵 (256 ビット鍵) 50

10 ハッシュ関数での利用 (256 ビット鍵) 51
10.1 ハッシュ関数と衝突困難性 51
10.2 256 ビット鍵 HyRAL による Davies-Meyer 方式圧縮関数の解析 . . . 51
10.3 256 ビット鍵 HyRAL を用いた Merkle-Damgård 方式によるハッシュ関数の解析 52
10.4 本章のまとめ 53

11 関連鍵攻撃 (256 ビット鍵) 54
11.1 関連鍵攻撃の概要 54
11.2 256 ビット鍵 HyRAL に対する関連鍵攻撃 54

12 関連暗号攻撃 (129, 130, . . . , 256 ビット鍵) 56
12.1 関連暗号攻撃の概要 56
12.2 **129, 130, . . . , 256** ビット鍵 HyRAL に対する関連暗号攻撃 56

13 圧縮関数での利用 (129, 130, . . . , 256 ビット鍵) 58

14 その他の攻撃 59

15 まとめ 60

謝辞 61

参考文献 62

図 目 次

1 128 ビット鍵 HyRAL の全体構造. 1
2 256 ビット鍵 HyRAL の全体構造. 3
3 128 ビット鍵 HyRAL の鍵生成アルゴリズム KGA. 4
4 256 ビット鍵 HyRAL の鍵生成アルゴリズム KGA_1, KGA_2. 5
5 DPA (128 ビット鍵). 7
6 DPA (256 ビット鍵). 7
7 G_1 関数 (左), G_2 関数 (右). 8
8 F_1 関数 (左), F_2 関数 (右). 10
9 f_i 関数. 12
10 T_i 関数. 13
11 KGA_1, KGA_2 の差分の伝搬. 29
12 active f_i 関数の個数が 4 である差分パス. 33
13 KGA のはじめの 8 ラウンド. 38
14 Davies-Meyer 方式. ブロック暗号の三角は鍵入力を表す. 51
15 Merkle-Damgård 方式. 52

表 目 次

1 差分パスと active f_i 関数. 32
2 各入力差分に対する active f_i 関数の個数. 34
3 $\mathrm{DCP}^{\mathrm{KGA}}(\delta) > 2^{-128}$ となる δ の例と個数. 36
4 等価鍵導出アルゴリズムの実行結果. 45
5 OK_1, OK_2 の導出における $I_8^{(5)}$ の探索範囲と計算時間. 46

1 HyRALの仕様

本章ではHyRALの仕様を示す.

HyRALはブロック長128ビット，鍵長は$128, 129, \ldots, 256$ビットをサポートするブロック暗号である[15, 17]. HyRALは鍵長が128ビットの場合と，$129, 130, \ldots, 256$ビットの場合で構造が異なり，以降，鍵長がkビットのHyRALを「kビット鍵HyRAL」と表記する.

kビット鍵HyRALは内部で

- 鍵生成アルゴリズム (KGA: Key Generation Algorithm)
- 鍵割り当てアルゴリズム (KAA: Key Assignment Algorithm)
- データ撹拌アルゴリズム (DPA: Data Processing Algorithm)

を用いる.

128ビット鍵HyRALの全体構造を図1に示す.

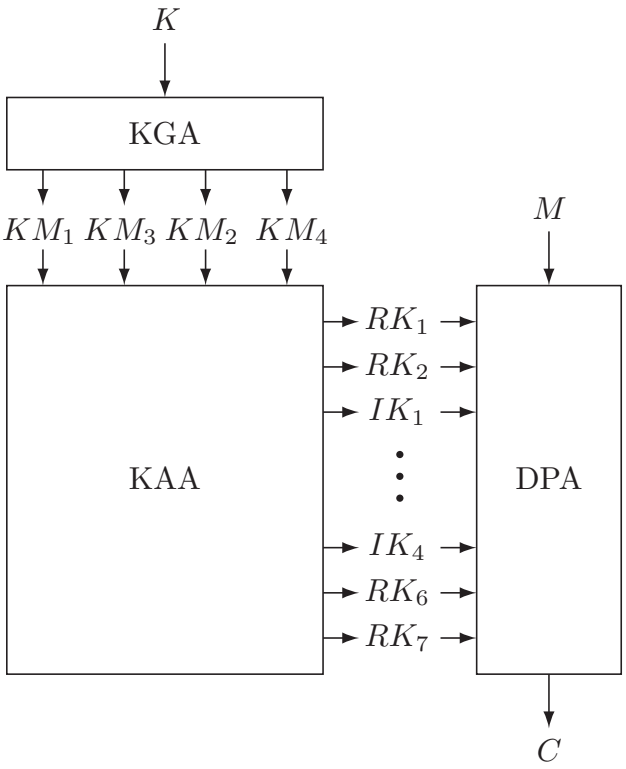

図1: 128ビット鍵HyRALの全体構造.

入力は秘密鍵$K \in \{0,1\}^{128}$と平文$M \in \{0,1\}^{128}$であり，出力は暗号文$C \in \{0,1\}^{128}$である.

秘密鍵$K \in \{0,1\}^{128}$と平文$M \in \{0,1\}^{128}$に対し，以下のように暗号化を行う.

1. 鍵生成アルゴリズム KGA に K を入力し，中間データ

$$KM = (KM_1, KM_3, KM_2, KM_4) \leftarrow \mathrm{KGA}(K)$$

を生成する．$KM_i \in \{0,1\}^{128}$ である．

2. KM を鍵割り当てアルゴリズム KAA に入力し，

$$(RK_1, \ldots, RK_7, IK_1, \ldots, IK_4) \leftarrow \mathrm{KAA}(KM)$$

を計算する．$RK_i, IK_i \in \{0,1\}^{128}$ である．

3. $RK_1, \ldots, RK_7, IK_1, \ldots, IK_4$ 及び平文 M をデータ撹拌アルゴリズム DPA に入力し，暗号文

$$C \leftarrow \mathrm{DPA}(RK_1, \ldots, RK_7, IK_1, \ldots, IK_4, M)$$

を計算し，C を出力する．

次に256ビット鍵HyRALの全体構造を図2に示す．入力は秘密鍵$K \in \{0,1\}^{256}$と平文$M \in \{0,1\}^{128}$であり，出力は暗号文$C \in \{0,1\}^{128}$である．256ビット鍵HyRALでは鍵生成アルゴリズムを2回使用し，それぞれ KGA_1, KGA_2 と表記する．

秘密鍵 $K \in \{0,1\}^{256}$ と平文 $M \in \{0,1\}^{128}$ に対し，以下のように暗号化を行う．

1. 秘密鍵 K の上位128ビットを OK_1 とし，下位128ビットを OK_2 とする．すなわち，$(OK_1, OK_2) \leftarrow K$ とする．
2. 鍵生成アルゴリズム KGA_1, KGA_2 に，それぞれ OK_1, OK_2 を入力し，中間データ $(Y_4, Y_5, Y_6, Y_7) \leftarrow \mathrm{KGA}_1(OK_1)$, $(Z_4, Z_5, Z_6, Z_7) \leftarrow \mathrm{KGA}_2(OK_2)$ を生成する．$Y_i, Z_i \in \{0,1\}^{128}$ である．
3. $KM = (KM_1, KM_3, KM_2, KM_4) \leftarrow (Y_4 \oplus Z_4, Y_5 \oplus Z_5, Y_6 \oplus Z_6, Y_7 \oplus Z_7)$ を計算する．$KM_i \in \{0,1\}^{128}$ である．
4. KM を鍵割り当てアルゴリズム KAA に入力し，

$$(RK_1, \ldots, RK_9, IK_1, \ldots, IK_6) \leftarrow \mathrm{KAA}(KM)$$

を計算する．$RK_i, IK_i \in \{0,1\}^{128}$ である．

5. $RK_1, \ldots, RK_9, IK_1, \ldots, IK_6$ 及び平文 M をデータ撹拌アルゴリズム DPA に入力し，暗号文

$$C \leftarrow \mathrm{DPA}(RK_1, \ldots, RK_9, IK_1, \ldots, IK_6, M)$$

を計算し，C を出力する．

k ビット鍵 HyRAL $(k = 129, 130, \ldots, 255)$ の入力は秘密鍵 $K \in \{0,1\}^k$ と平文 $M \in \{0,1\}^{128}$ であり，出力は暗号文 $C \in \{0,1\}^{128}$ である．256 ビット鍵 HyRAL と OK_1 と OK_2 の導出以外は同一である．OK_1 は K の上位 128 ビットであり，OK_2 は K の下位 $k-128$ ビットに 0 を $256-k$ ビット連結したビット列である．

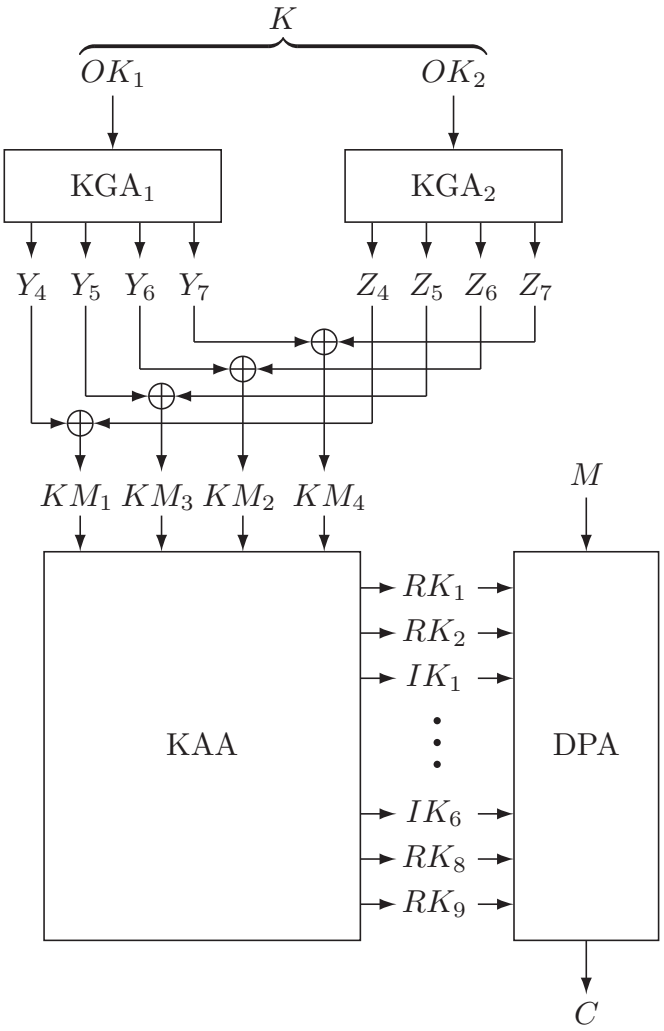

図 2: 256 ビット鍵 HyRAL の全体構造.

1.1 鍵生成アルゴリズム (128 ビット鍵): KGA

128 ビット鍵 HyRAL の鍵生成アルゴリズムを図 3 に示す.

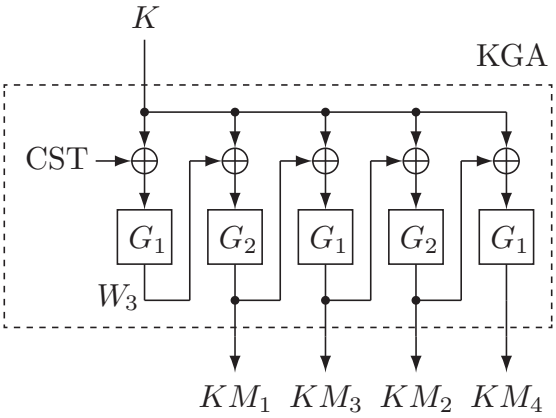

図 3: 128 ビット鍵 HyRAL の鍵生成アルゴリズム KGA.

KGA は内部で入出力 128 ビットの拡大関数である G_1, G_2 関数，及び以下の 128 ビットの定数 CST を用いる．0x は 16 進数表記であることを表す.

$$\mathrm{CST} = \texttt{0x628ccda03b1565c13bad2d4fb8806ac5}$$

KGA は秘密鍵 $K \in \{0,1\}^{128}$ を入力とし，合計 512 ビット (128 ビット $\times 4$) の中間データ (KM_1, KM_3, KM_2, KM_4) を以下のように生成する.

1. $W_3 \leftarrow G_1(K \oplus \mathrm{CST})$
2. $KM_1 \leftarrow G_2(K \oplus W_3)$
3. $KM_3 \leftarrow G_1(K \oplus KM_1)$
4. $KM_2 \leftarrow G_2(K \oplus KM_3)$
5. $KM_4 \leftarrow G_1(K \oplus KM_2)$
6. (KM_1, KM_3, KM_2, KM_4) を出力する.

1.2 鍵生成アルゴリズム ($129, 130, \ldots, 256$ ビット鍵): $\mathrm{KGA}_1, \mathrm{KGA}_2$

$129, 130, \ldots, 256$ ビット鍵 HyRAL の鍵生成アルゴリズムを図 4 に示す.

$\mathrm{KGA}_1, \mathrm{KGA}_2$ は内部で入出力 128 ビットの拡大関数である G_1, G_2 関数を用いる. KGA_1 と KGA_2 は内部で使用する定数のみが異なり，それぞれ以下の 128 ビットの定数 CST_1, CST_2 を用いる.

$$\begin{cases} \mathrm{CST}_1 = \texttt{0xf9251a2365cd3c2e8066cbbbfe316b7b} \\ \mathrm{CST}_2 = \texttt{0x5de28625656b71ff9ffb1e12eef127f5} \end{cases}$$

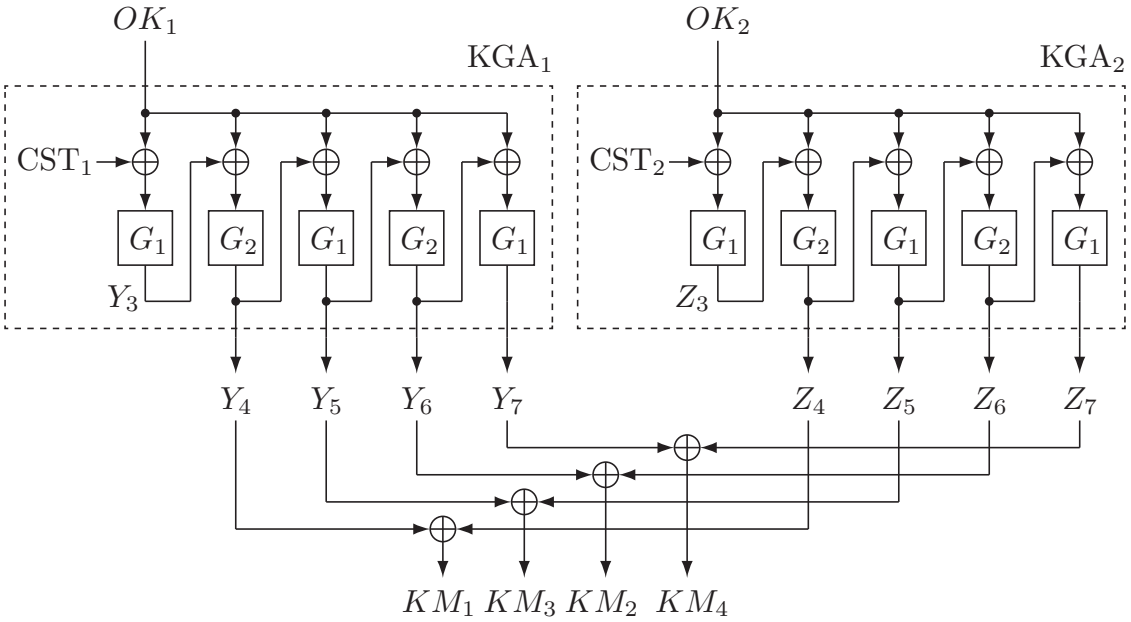

図 4: 256 ビット鍵 HyRAL の鍵生成アルゴリズム KGA$_1$, KGA$_2$.

KGA$_1$ は $OK_1 \in \{0,1\}^{128}$ を入力とし，合計 512 ビットの中間データ (Y_4, Y_5, Y_6, Y_7) を以下のように生成する．

1. $Y_3 \leftarrow G_1(OK_1 \oplus \mathrm{CST}_1)$
2. $Y_4 \leftarrow G_2(OK_1 \oplus Y_3)$
3. $Y_5 \leftarrow G_1(OK_1 \oplus Y_4)$
4. $Y_6 \leftarrow G_2(OK_1 \oplus Y_5)$
5. $Y_7 \leftarrow G_1(OK_1 \oplus Y_6)$
6. (Y_4, Y_5, Y_6, Y_7) を出力する．

また KGA$_2$ は OK_2 を入力とし，合計 512 ビットの中間データ (Z_4, Z_5, Z_6, Z_7) を出力する．KGA$_2$ では，KGA$_1$ における $(OK_1, \mathrm{CST}_1, Y_3, Y_4, Y_5, Y_6, Y_7)$ をそれぞれ $(OK_2, \mathrm{CST}_2, Z_3, Z_4, Z_5, Z_6, Z_7)$ と置き換え，以下のように (Z_4, Z_5, Z_6, Z_7) を生成する．

1. $Z_3 \leftarrow G_1(OK_2 \oplus \mathrm{CST}_2)$
2. $Z_4 \leftarrow G_2(OK_2 \oplus Z_3)$
3. $Z_5 \leftarrow G_1(OK_2 \oplus Z_4)$
4. $Z_6 \leftarrow G_2(OK_2 \oplus Z_5)$
5. $Z_7 \leftarrow G_1(OK_2 \oplus Z_6)$
6. (Z_4, Z_5, Z_6, Z_7) を出力する．

1.3 データ撹拌関数 (128 ビット鍵): DPA

128 ビット鍵 HyRAL のデータ撹拌関数 DPA は鍵割り当てアルゴリズム KAA の出力 $RK_1, \ldots, RK_7, IK_1, \ldots, IK_4$，平文 $M \in \{0,1\}^{128}$ を入力とし，暗号文 $C \in \{0,1\}^{128}$ を生成する．内部では入出力 128 ビットの拡大関数である G_1, G_2 関数，及び入力 256 ビット，出力 128 ビットの拡大関数である F_1, F_2 関数を用いる．DPA の処理手順を以下に示す．

1. $X \leftarrow M$
2. $X \leftarrow G_1(X \oplus RK_1)$
3. $X \leftarrow F_2(IK_1, X \oplus RK_2)$
4. $X \leftarrow F_2(IK_2, X \oplus RK_3)$
5. $X \leftarrow F_1(IK_3, X \oplus RK_4)$
6. $X \leftarrow F_1(IK_4, X \oplus RK_5)$
7. $X \leftarrow G_2(X \oplus RK_6)$
8. $C \leftarrow X \oplus RK_7$
9. C を出力する．

DPA を図 5 に図示する．

1.4 データ撹拌関数 ($129, 130, \ldots, 256$ ビット鍵): DPA

$129, 130, \ldots, 256$ ビット鍵 HyRAL のデータ撹拌関数 DPA は鍵割り当てアルゴリズム KAA の出力 $RK_1, \ldots, RK_9, IK_1, \ldots, IK_6$，平文 $M \in \{0,1\}^{128}$ を入力とし，暗号文 $C \in \{0,1\}^{128}$ を生成する．128 ビット鍵の場合と同様に，内部では入出力 128 ビットの拡大関数である G_1, G_2 関数，及び入力 256 ビット，出力 128 ビットの拡大関数である F_1, F_2 関数を用いる．DPA の処理手順を以下に示す．

1. $X \leftarrow M$
2. $X \leftarrow G_1(X \oplus RK_1)$
3. $X \leftarrow F_2(IK_1, X \oplus RK_2)$
4. $X \leftarrow F_2(IK_2, X \oplus RK_3)$
5. $X \leftarrow F_2(IK_3, X \oplus RK_4)$
6. $X \leftarrow F_1(IK_4, X \oplus RK_5)$
7. $X \leftarrow F_1(IK_5, X \oplus RK_6)$
8. $X \leftarrow F_1(IK_6, X \oplus RK_7)$

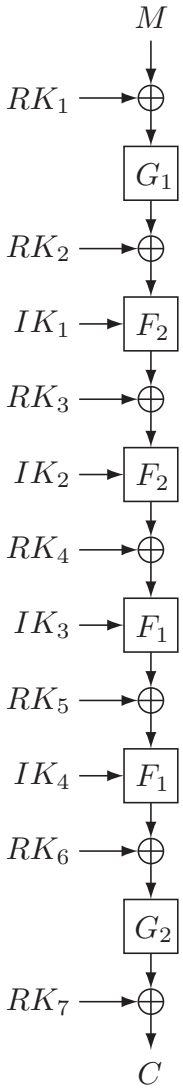

図 5: DPA (128 ビット鍵).

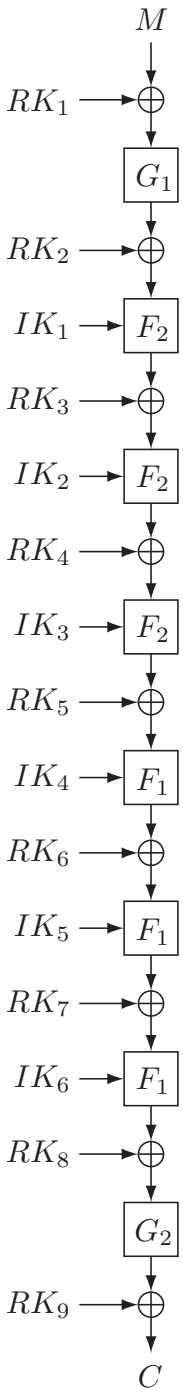

図 6: DPA (256 ビット鍵).

9. $X \leftarrow G_2(X \oplus RK_8)$
10. $C \leftarrow X \oplus RK_9$
11. C を出力する．

256 ビット鍵 HyRAL の DPA を図 6 に示す．

1.5 拡大関数: G_1, G_2 関数

拡大関数である G_1, G_2 関数を図 7 に示す．

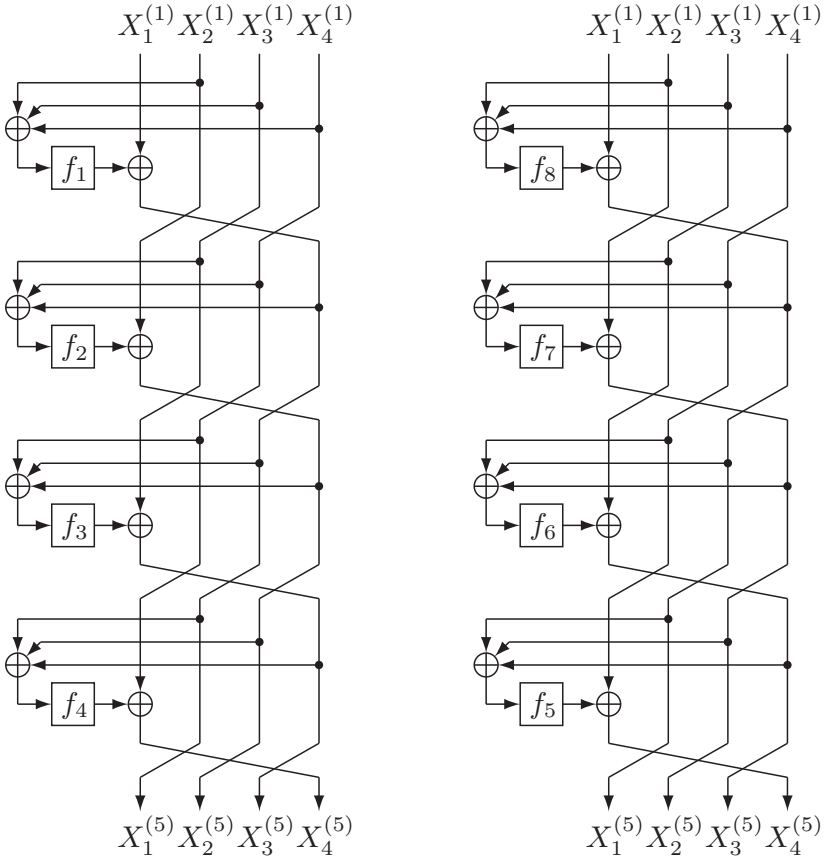

図 7: G_1 関数 (左)，G_2 関数 (右)．

G_1, G_2 関数の入出力は 128 ビットであり，ともに 4 ラウンド，4 系列の一般化 Feistel 型構造である．G_1 関数は内部で 32 ビット入出力の基本関数である f_1, f_2, f_3, f_4 関数を用い，G_2 関数は f_5, f_6, f_7, f_8 関数を用いる．G_1 関数は $X \in \{0,1\}^{128}$ を入力とし，以下のように出力を計算する．

1. X を 32 ビットに分割し，$(X_1^{(1)}, X_2^{(1)}, X_3^{(1)}, X_4^{(1)}) = X$ とする．

2. $r = 1, 2, 3, 4$ に対し，次式を計算する．

$$\begin{cases} X_1^{(r+1)} \leftarrow X_2^{(r)} \\ X_2^{(r+1)} \leftarrow X_3^{(r)} \\ X_3^{(r+1)} \leftarrow X_4^{(r)} \\ X_4^{(r+1)} \leftarrow X_1^{(r)} \oplus f_r(X_2^{(r)} \oplus X_3^{(r)} \oplus X_4^{(r)}) \end{cases}$$

3. $(X_1^{(5)}, X_2^{(5)}, X_3^{(5)}, X_4^{(5)})$ を出力する．

ステップ2の各手順を1ラウンドと呼ぶ．G_2 関数では，G_1 関数における (f_1, f_2, f_3, f_4) をそれぞれ (f_8, f_7, f_6, f_5) と置き換える．

1.6 拡大関数: F_1, F_2 関数

拡大関数である F_1, F_2 関数を図 8 に示す．

F_1, F_2 関数の入力は 128 ビットの $X = (X_1^{(1)}, X_2^{(1)}, X_3^{(1)}, X_4^{(1)})$ 及び 128 ビットの $IK_i = (IK_{i,1}, IK_{i,2}, IK_{i,3}, IK_{i,4})$ の計 256 ビットであり，出力は 128 ビットである．F_1, F_2 関数はともに 4 ラウンド，4 系列の一般化 Feistel 型構造である．F_1 関数は内部で 32 ビット入出力の基本関数である f_1, f_2, f_3, f_4 関数を用い，F_2 関数は f_5, f_6, f_7, f_8 関数を用いる．F_1 関数は以下のように出力を計算する．

1. 入力 $X \in \{0,1\}^{128}$ を 32 ビットに分割し，$(X_1^{(1)}, X_2^{(1)}, X_3^{(1)}, X_4^{(1)}) = X$ とする．
2. 次式を計算する．

$$\begin{cases} X_1^{(2)} \leftarrow X_4^{(1)} \\ X_2^{(2)} \leftarrow X_1^{(1)} \\ X_3^{(2)} \leftarrow X_2^{(1)} \oplus f_3(f_4(X_4^{(1)}) \oplus X_1^{(1)} \oplus IK_{i,4}) \\ X_4^{(2)} \leftarrow X_3^{(1)} \end{cases}$$

$$\begin{cases} X_1^{(3)} \leftarrow X_4^{(2)} \\ X_2^{(3)} \leftarrow X_1^{(2)} \\ X_3^{(3)} \leftarrow X_2^{(2)} \oplus f_1(f_2(X_4^{(2)}) \oplus X_1^{(2)} \oplus IK_{i,3}) \\ X_4^{(3)} \leftarrow X_3^{(2)} \end{cases}$$

$$\begin{cases} X_1^{(4)} \leftarrow X_4^{(3)} \\ X_2^{(4)} \leftarrow X_1^{(3)} \\ X_3^{(4)} \leftarrow X_2^{(3)} \oplus f_4(f_3(X_4^{(3)}) \oplus X_1^{(3)} \oplus IK_{i,2}) \\ X_4^{(4)} \leftarrow X_3^{(3)} \end{cases}$$

$$\begin{cases} X_1^{(5)} \leftarrow X_4^{(4)} \\ X_2^{(5)} \leftarrow X_1^{(4)} \\ X_3^{(5)} \leftarrow X_2^{(4)} \oplus f_2(f_1(X_4^{(4)}) \oplus X_1^{(4)} \oplus IK_{i,1}) \\ X_4^{(5)} \leftarrow X_3^{(4)} \end{cases}$$

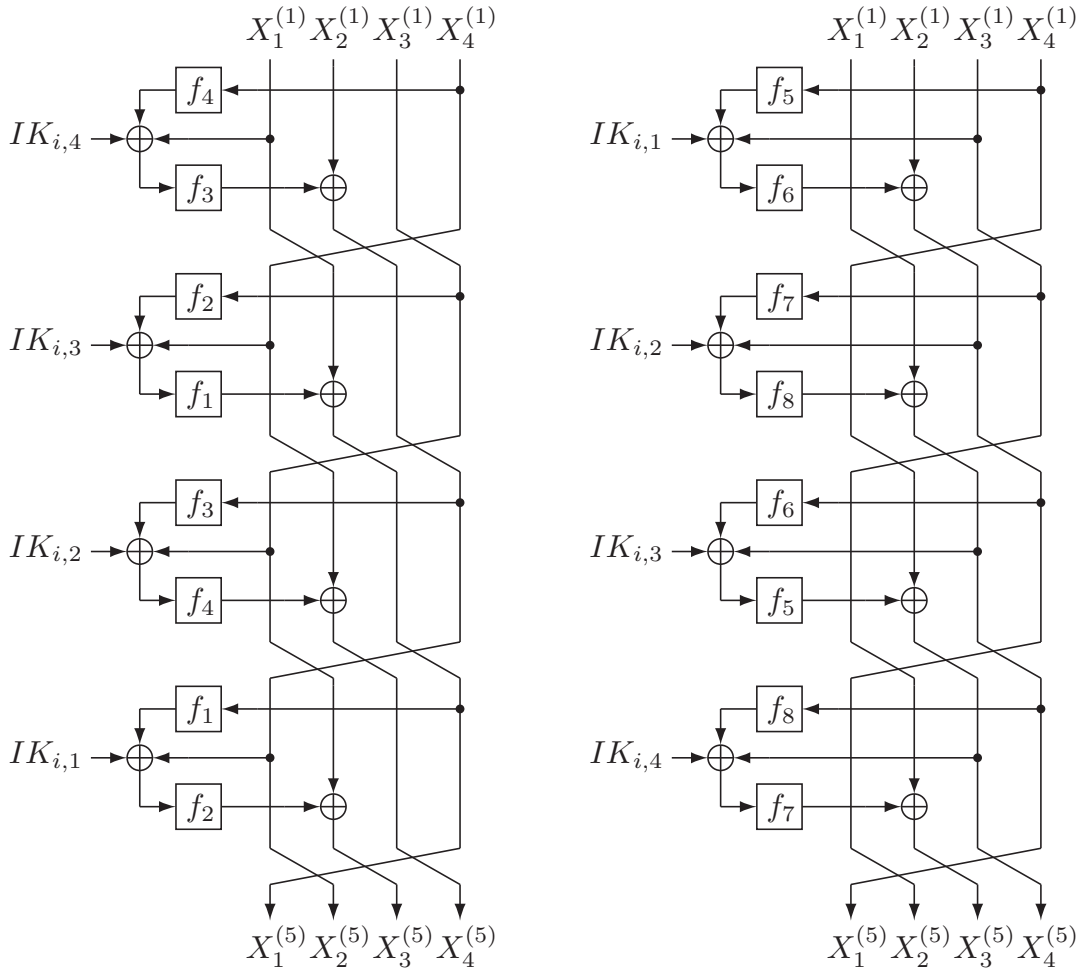

図 8: F_1 関数 (左)，F_2 関数 (右).

3. $(X_1^{(5)}, X_2^{(5)}, X_3^{(5)}, X_4^{(5)})$ を出力する．

ステップ 2 の各手順を 1 ラウンドと呼ぶ．F_2 関数では，以下のように出力を計算する．

1. 入力 $X \in \{0,1\}^{128}$ を 32 ビットに分割し，$(X_1^{(1)}, X_2^{(1)}, X_3^{(1)}, X_4^{(1)}) = X$ とする．
2. 次式を計算する．

$$
\begin{cases}
X_1^{(2)} \leftarrow X_4^{(1)} \\
X_2^{(2)} \leftarrow X_1^{(1)} \\
X_3^{(2)} \leftarrow X_2^{(1)} \oplus f_6(f_5(X_4^{(1)}) \oplus X_3^{(1)} \oplus IK_{i,1}) \\
X_4^{(2)} \leftarrow X_3^{(1)}
\end{cases}
$$

$$
\begin{cases}
X_1^{(3)} \leftarrow X_4^{(2)} \\
X_2^{(3)} \leftarrow X_1^{(2)} \\
X_3^{(3)} \leftarrow X_2^{(2)} \oplus f_8(f_7(X_4^{(2)}) \oplus X_3^{(2)} \oplus IK_{i,2}) \\
X_4^{(3)} \leftarrow X_3^{(2)}
\end{cases}
$$

$$
\begin{cases}
X_1^{(4)} \leftarrow X_4^{(3)} \\
X_2^{(4)} \leftarrow X_1^{(3)} \\
X_3^{(4)} \leftarrow X_2^{(3)} \oplus f_5(f_6(X_4^{(3)}) \oplus X_3^{(3)} \oplus IK_{i,3}) \\
X_4^{(4)} \leftarrow X_3^{(3)}
\end{cases}
$$

$$
\begin{cases}
X_1^{(5)} \leftarrow X_4^{(4)} \\
X_2^{(5)} \leftarrow X_1^{(4)} \\
X_3^{(5)} \leftarrow X_2^{(4)} \oplus f_7(f_8(X_4^{(4)}) \oplus X_3^{(4)} \oplus IK_{i,4}) \\
X_4^{(5)} \leftarrow X_3^{(4)}
\end{cases}
$$

3. $(X_1^{(5)}, X_2^{(5)}, X_3^{(5)}, X_4^{(5)})$ を出力する．

F_2 関数においても，ステップ 2 の各手順を 1 ラウンドと呼ぶ．

1.7 基本関数: $f_1, \ldots, f_8$ 関数

基本関数である $f_1, \ldots, f_8$ 関数を図 9 に示す．

$f_1, \ldots, f_8$ 関数は入出力が 32 ビットであり，$\{0,1\}^{32}$ 上の全単射である．f_i 関数はバイト転置 T_i，S 層，P 層，定数 xor からなる．T_i は i に依存したバイト転置を行う．S 層では，8 ビット入出力の S-box S を各バイトに適用し，P 層では MDS 行列の乗算を行う．32 ビットの入力 $I \in \{0,1\}^{32}$ に対し，以下の手順で出力を計算する．

1. I をバイトに分割し，$(x_1, x_2, x_3, x_4) = I$ とする．
2. $(y_1, y_2, y_3, y_4) \leftarrow T_i(x_1, x_2, x_3, x_4)$

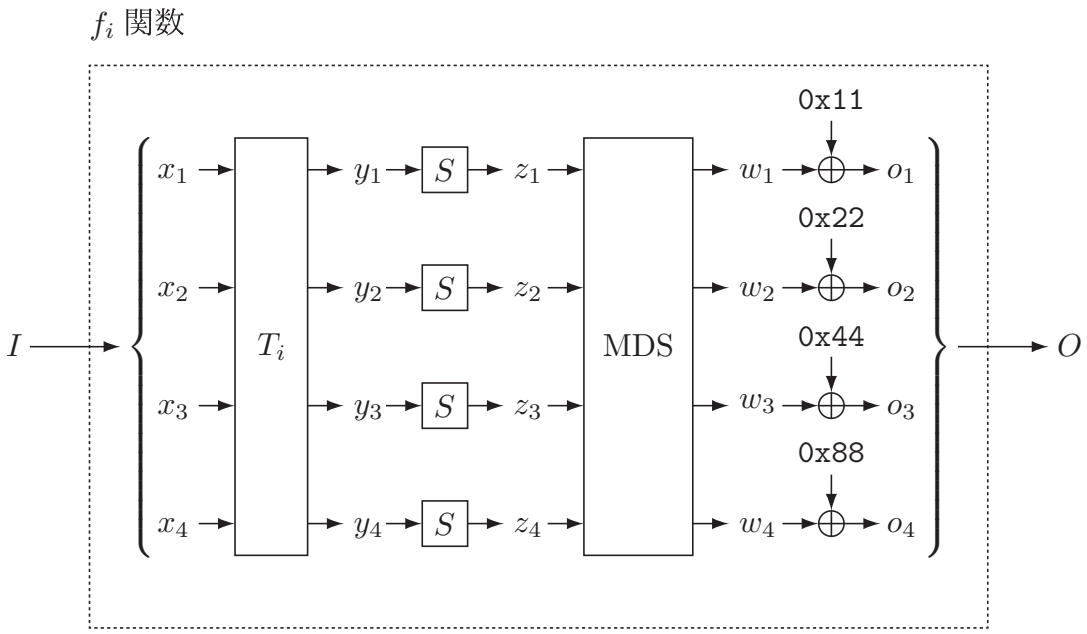

図 9: f_i 関数.

3. $(z_1, z_2, z_3, z_4) \leftarrow (S(y_1), S(y_2), S(y_3), S(y_4))$
4. (w_1, w_2, w_3, w_4) を次式で計算する.

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} \texttt{0x03} & \texttt{0x03} & \texttt{0x02} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x02} & \texttt{0x02} \\ \texttt{0x07} & \texttt{0x03} & \texttt{0x01} & \texttt{0x02} \\ \texttt{0x07} & \texttt{0x04} & \texttt{0x05} & \texttt{0x03} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix}$$

   ただし，演算は規約多項式 $x^8 + x^4 + x^3 + x + 1$ で定義される拡大体 $\mathrm{GF}(2^8)$ 上で行う.

5. $(o_1, o_2, o_3, o_4) \leftarrow (w_1, w_2, w_3, w_4) \oplus (\texttt{0x11}, \texttt{0x22}, \texttt{0x44}, \texttt{0x88})$ とする.
6. $O \leftarrow (o_1, o_2, o_3, o_4)$ を出力する.

f_i 関数で用いられる転置 T_i は以下のように定義される.

$$T_i(x_1, x_2, x_3, x_4) = \begin{cases} (x_1, x_2, x_3, x_4) & i = 1 \\ (x_2, x_3, x_4, x_1) & i = 2 \\ (x_3, x_4, x_1, x_2) & i = 3 \\ (x_4, x_1, x_2, x_3) & i = 4 \\ (x_4, x_3, x_2, x_1) & i = 5 \\ (x_3, x_2, x_1, x_4) & i = 6 \\ (x_2, x_1, x_4, x_3) & i = 7 \\ (x_1, x_4, x_3, x_2) & i = 8 \end{cases}$$

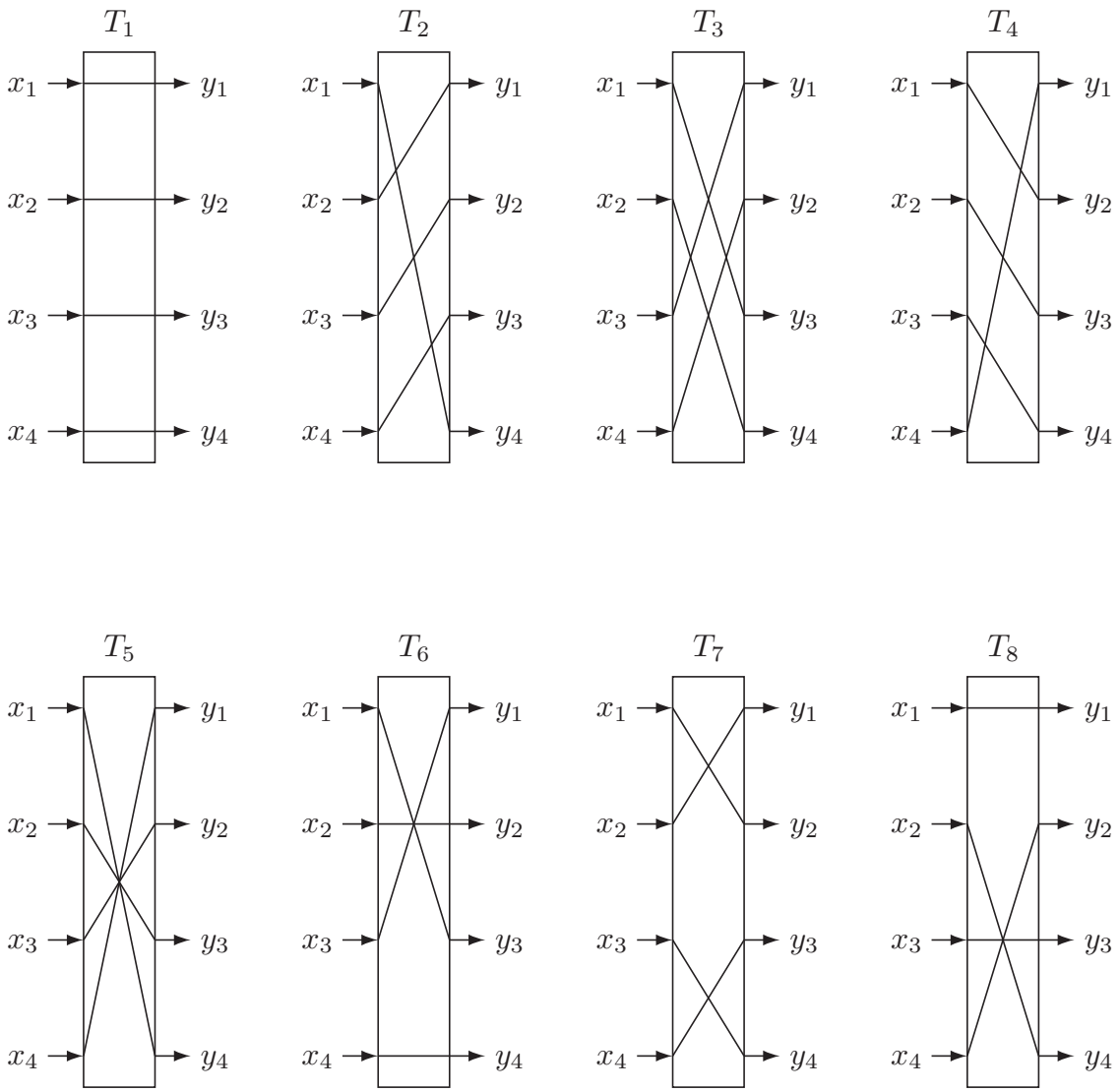

図 10: T_i 関数.

転置 T_i を図 10 に示す．

S-box S は GF(2) 上のアフィン変換と GF(2^8) 上の逆関数の合成関数として定義される．S-box の入出力を以下に示す．

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	5e	d3	af	36	43	a6	49	33	93	3b	21	91	df	47	f4
1	b6	70	06	d0	81	82	fa	a1	10	b5	3c	ba	97	85	b7	79
2	ed	5c	ca	05	87	bf	24	4c	51	ec	17	61	22	f0	3e	18
3	a7	64	13	ab	e9	09	25	54	2d	31	69	f5	37	67	fe	1d
4	0b	28	a3	2f	e4	0f	d4	da	1b	fc	e6	ac	53	04	27	a9
5	94	8b	d5	c4	90	6b	f8	9d	c5	db	ea	e2	ae	63	07	7a
6	5b	23	34	38	03	8c	46	68	cd	1a	1c	41	7d	a0	9c	dd
7	08	4e	e3	d7	1e	b3	50	5d	c6	0e	ad	cf	d6	eb	0d	b1
8	fb	7c	c3	2e	65	48	b8	8f	ce	e7	62	d2	12	4a	c8	26
9	a5	8e	3d	76	86	57	bc	bd	11	75	71	78	1f	ef	e0	0c
a	de	6a	6d	32	84	72	8a	d8	f9	dc	9a	89	9f	88	14	2a
b	9b	9e	d9	95	b9	a4	02	f7	96	73	56	be	7f	80	7e	83
c	00	01	f6	8d	7b	d1	52	cb	b0	e1	c7	e5	29	c0	4f	e8
d	58	3f	cc	fd	ee	b2	40	ff	99	2b	5f	60	aa	4b	b4	74
e	2c	45	6c	92	66	42	39	f3	77	bb	19	59	20	6f	35	f2
f	c1	0a	15	98	a2	c2	44	30	55	4d	c9	a8	5a	f1	6e	3a

ただし，各値は 16 進数で表記されており，入力 y を 2 桁の 16 進数とみなし，1 桁目を i，2 桁目を j とするとき，i 行 j 列にある値が出力される．例えば，S(0x12) は 0x06 である．

1.8 鍵割り当てアルゴリズム (128 ビット鍵): KAA

鍵割り当てアルゴリズム KAA は合計 512 ビットの (KM_1, KM_3, KM_2, KM_4) を入力とし，$RK_1, \ldots, RK_7, IK_1, \ldots, IK_4$ を出力する．以降，KM_i を 32 ビットに分割したビット列を $(KM_{i,1}, KM_{i,2}, KM_{i,3}, KM_{i,4})$ と表記し，同様に RK_i, IK_i の 32 ビット分割をそれぞれ $(RK_{i,1}, RK_{i,2}, RK_{i,3}, RK_{i,4})$, $(IK_{i,1}, IK_{i,2}, IK_{i,3}, IK_{i,4})$ と表記する．

1. $(RK_1, \ldots, RK_7)$ を次式により計算する．

$$\begin{cases} RK_{1,1} \leftarrow KM_{3,4} \oplus KM_{4,1} \\ RK_{1,2} \leftarrow KM_{3,3} \oplus KM_{4,2} \\ RK_{1,3} \leftarrow KM_{3,2} \oplus KM_{4,3} \\ RK_{1,4} \leftarrow KM_{3,1} \oplus KM_{4,4} \end{cases}$$

$$
\begin{cases}
RK_{2,1} \leftarrow KM_{1,1} \oplus KM_{2,2} \\
RK_{2,2} \leftarrow KM_{1,4} \oplus KM_{2,3} \\
RK_{2,3} \leftarrow KM_{1,3} \oplus KM_{2,4} \\
RK_{2,4} \leftarrow KM_{1,2} \oplus KM_{2,1}
\end{cases}
\begin{cases}
RK_{3,1} \leftarrow KM_{3,1} \oplus KM_{4,3} \\
RK_{3,2} \leftarrow KM_{3,2} \oplus KM_{4,4} \\
RK_{3,3} \leftarrow KM_{3,3} \oplus KM_{4,1} \\
RK_{3,4} \leftarrow KM_{3,4} \oplus KM_{4,2}
\end{cases}
\begin{cases}
RK_{4,1} \leftarrow KM_{1,1} \oplus KM_{4,4} \\
RK_{4,2} \leftarrow KM_{1,2} \oplus KM_{4,3} \\
RK_{4,3} \leftarrow KM_{1,3} \oplus KM_{4,2} \\
RK_{4,4} \leftarrow KM_{1,4} \oplus KM_{4,1}
\end{cases}
\begin{cases}
RK_{5,1} \leftarrow KM_{1,1} \oplus KM_{2,3} \\
RK_{5,2} \leftarrow KM_{1,2} \oplus KM_{2,4} \\
RK_{5,3} \leftarrow KM_{1,3} \oplus KM_{2,1} \\
RK_{5,4} \leftarrow KM_{1,4} \oplus KM_{2,2}
\end{cases}
\begin{cases}
RK_{6,1} \leftarrow KM_{3,1} \oplus KM_{4,2} \\
RK_{6,2} \leftarrow KM_{3,4} \oplus KM_{4,3} \\
RK_{6,3} \leftarrow KM_{3,3} \oplus KM_{4,4} \\
RK_{6,4} \leftarrow KM_{3,2} \oplus KM_{4,1}
\end{cases}
\begin{cases}
RK_{7,1} \leftarrow KM_{1,4} \oplus KM_{2,1} \\
RK_{7,2} \leftarrow KM_{1,3} \oplus KM_{2,2} \\
RK_{7,3} \leftarrow KM_{1,2} \oplus KM_{2,3} \\
RK_{7,4} \leftarrow KM_{1,1} \oplus KM_{2,4}
\end{cases}
$$

2. $(IK_1, \ldots, IK_4)$ を次式により計算する.

$$
\begin{cases}
IK_{1,1} \leftarrow KM_{1,2} \oplus KM_{4,1} \\
IK_{1,2} \leftarrow KM_{1,3} \oplus KM_{4,4} \\
IK_{1,3} \leftarrow KM_{1,4} \oplus KM_{4,3} \\
IK_{1,4} \leftarrow KM_{1,1} \oplus KM_{4,2}
\end{cases}
\begin{cases}
IK_{2,1} \leftarrow KM_{3,4} \oplus KM_{4,4} \\
IK_{2,2} \leftarrow KM_{3,1} \oplus KM_{4,1} \\
IK_{2,3} \leftarrow KM_{3,2} \oplus KM_{4,2} \\
IK_{2,4} \leftarrow KM_{3,3} \oplus KM_{4,3}
\end{cases}
$$

$$
\begin{cases}
IK_{3,1} \leftarrow KM_{1,4} \oplus KM_{2,4} \\
IK_{3,2} \leftarrow KM_{1,1} \oplus KM_{2,1} \\
IK_{3,3} \leftarrow KM_{1,2} \oplus KM_{2,2} \\
IK_{3,4} \leftarrow KM_{1,3} \oplus KM_{2,3}
\end{cases}
\begin{cases}
IK_{4,1} \leftarrow KM_{1,3} \oplus KM_{4,1} \\
IK_{4,2} \leftarrow KM_{1,4} \oplus KM_{4,2} \\
IK_{4,3} \leftarrow KM_{1,1} \oplus KM_{4,3} \\
IK_{4,4} \leftarrow KM_{1,2} \oplus KM_{4,4}
\end{cases}
$$

3. $(RK_1, \ldots, RK_7, IK_1, \ldots, IK_4)$ を出力する．

1.9 鍵割り当てアルゴリズム ($129, 130, \ldots, 256$ ビット鍵): KAA

$129, 130, \ldots, 256$ ビット鍵 HyRAL の鍵割り当てアルゴリズム KAA は合計 512 ビットの (KM_1, KM_3, KM_2, KM_4) を入力とし，$RK_1, \ldots, RK_9, IK_1, \ldots, IK_6$ を出力する．128 ビット鍵の場合と同様に，KM_i を 32 ビットに分割したビット列を $(KM_{i,1}, KM_{i,2}, KM_{i,3}, KM_{i,4})$ と表記し，同様に RK_i, IK_i の 32 ビット分割をそれぞれ $(RK_{i,1}, RK_{i,2}, RK_{i,3}, RK_{i,4})$, $(IK_{i,1}, IK_{i,2}, IK_{i,3}, IK_{i,4})$ と表記する．

1. 次式を計算する．

$$
\begin{cases}
RK_{1,1} \leftarrow KM_{3,4} \oplus KM_{4,1} \\
RK_{1,2} \leftarrow KM_{3,3} \oplus KM_{4,2} \\
RK_{1,3} \leftarrow KM_{3,2} \oplus KM_{4,3} \\
RK_{1,4} \leftarrow KM_{3,1} \oplus KM_{4,4}
\end{cases}
\begin{cases}
RK_{2,1} \leftarrow KM_{1,1} \oplus KM_{2,2} \\
RK_{2,2} \leftarrow KM_{1,4} \oplus KM_{2,3} \\
RK_{2,3} \leftarrow KM_{1,3} \oplus KM_{2,4} \\
RK_{2,4} \leftarrow KM_{1,2} \oplus KM_{2,1}
\end{cases}
\begin{cases}
RK_{3,1} \leftarrow KM_{3,1} \oplus KM_{4,3} \\
RK_{3,2} \leftarrow KM_{3,2} \oplus KM_{4,4} \\
RK_{3,3} \leftarrow KM_{3,3} \oplus KM_{4,1} \\
RK_{3,4} \leftarrow KM_{3,4} \oplus KM_{4,2}
\end{cases}
\begin{cases}
RK_{4,1} \leftarrow KM_{3,4} \oplus KM_{4,4} \\
RK_{4,2} \leftarrow KM_{3,1} \oplus KM_{4,1} \\
RK_{4,3} \leftarrow KM_{3,2} \oplus KM_{4,2} \\
RK_{4,4} \leftarrow KM_{3,3} \oplus KM_{4,3}
\end{cases}
$$

$$
\begin{cases}
RK_{5,1} \leftarrow KM_{1,1} \oplus KM_{4,4} \\
RK_{5,2} \leftarrow KM_{1,2} \oplus KM_{4,3} \\
RK_{5,3} \leftarrow KM_{1,3} \oplus KM_{4,2} \\
RK_{5,4} \leftarrow KM_{1,4} \oplus KM_{4,1}
\end{cases}
\begin{cases}
RK_{6,1} \leftarrow KM_{1,4} \oplus KM_{2,4} \\
RK_{6,2} \leftarrow KM_{1,1} \oplus KM_{2,1} \\
RK_{6,3} \leftarrow KM_{1,2} \oplus KM_{2,2} \\
RK_{6,4} \leftarrow KM_{1,3} \oplus KM_{2,3}
\end{cases}
\begin{cases}
RK_{7,1} \leftarrow KM_{1,1} \oplus KM_{2,3} \\
RK_{7,2} \leftarrow KM_{1,2} \oplus KM_{2,4} \\
RK_{7,3} \leftarrow KM_{1,3} \oplus KM_{2,1} \\
RK_{7,4} \leftarrow KM_{1,4} \oplus KM_{2,2}
\end{cases}
\begin{cases}
RK_{8,1} \leftarrow KM_{3,1} \oplus KM_{4,2} \\
RK_{8,2} \leftarrow KM_{3,4} \oplus KM_{4,3} \\
RK_{8,3} \leftarrow KM_{3,3} \oplus KM_{4,4} \\
RK_{8,4} \leftarrow KM_{3,2} \oplus KM_{4,1}
\end{cases}
\begin{cases}
RK_{9,1} \leftarrow KM_{1,4} \oplus KM_{2,1} \\
RK_{9,2} \leftarrow KM_{1,3} \oplus KM_{2,2} \\
RK_{9,3} \leftarrow KM_{1,2} \oplus KM_{2,3} \\
RK_{9,4} \leftarrow KM_{1,1} \oplus KM_{2,4}
\end{cases}
$$

2. $(IK_1, \ldots, IK_6)$ を次式により計算する.

$$
\begin{cases}
IK_{1,1} \leftarrow KM_{1,2} \oplus KM_{4,1} \\
IK_{1,2} \leftarrow KM_{1,3} \oplus KM_{4,4} \\
IK_{1,3} \leftarrow KM_{1,4} \oplus KM_{4,3} \\
IK_{1,4} \leftarrow KM_{1,1} \oplus KM_{4,2}
\end{cases}
\begin{cases}
IK_{2,1} \leftarrow KM_{2,4} \oplus KM_{3,1} \\
IK_{2,2} \leftarrow KM_{2,3} \oplus KM_{3,2} \\
IK_{2,3} \leftarrow KM_{2,2} \oplus KM_{3,3} \\
IK_{2,4} \leftarrow KM_{2,1} \oplus KM_{3,4}
\end{cases}
\begin{cases}
IK_{3,1} \leftarrow KM_{2,1} \oplus KM_{3,2} \\
IK_{3,2} \leftarrow KM_{2,4} \oplus KM_{3,3} \\
IK_{3,3} \leftarrow KM_{2,3} \oplus KM_{3,4} \\
IK_{3,4} \leftarrow KM_{2,2} \oplus KM_{3,1}
\end{cases}
$$

$$
\begin{cases}
IK_{4,1} \leftarrow KM_{2,1} \oplus KM_{3,3} \\
IK_{4,2} \leftarrow KM_{2,2} \oplus KM_{3,4} \\
IK_{4,3} \leftarrow KM_{2,3} \oplus KM_{3,1} \\
IK_{4,4} \leftarrow KM_{2,4} \oplus KM_{3,2}
\end{cases}
\begin{cases}
IK_{5,1} \leftarrow KM_{2,4} \oplus KM_{3,4} \\
IK_{5,2} \leftarrow KM_{2,1} \oplus KM_{3,1} \\
IK_{5,3} \leftarrow KM_{2,2} \oplus KM_{3,2} \\
IK_{5,4} \leftarrow KM_{2,3} \oplus KM_{3,3}
\end{cases}
\begin{cases}
IK_{6,1} \leftarrow KM_{1,3} \oplus KM_{4,1} \\
IK_{6,2} \leftarrow KM_{1,4} \oplus KM_{4,2} \\
IK_{6,3} \leftarrow KM_{1,1} \oplus KM_{4,3} \\
IK_{6,4} \leftarrow KM_{1,2} \oplus KM_{4,4}
\end{cases}
$$

3. $(RK_1, \ldots, RK_9, IK_1, \ldots, IK_6)$ を出力する.

2　差分攻撃 (128, 192, 256 ビット鍵)

2.1　差分攻撃の概要

Biham と Shamir による差分攻撃はブロック暗号に対する汎用的な攻撃法であり [7, 8]，入力を変化させ差分を与えたとき，その出力の変化の分布の偏りを利用する攻撃である．まず [19] に沿って差分攻撃の概要を示す．

入出力が n ビットの関数 $f(x) : \{0,1\}^n \to \{0,1\}^n$ を考える．入力差分 Δx と出力差分 Δy が与えられたときの f の差分確率 $\mathrm{DP}^f(\Delta x, \Delta y)$ は以下のように定義される．

$$\mathrm{DP}^f(\Delta x, \Delta y) = \frac{\#\{x \in \{0,1\}^n \mid f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^n}$$

f の最大差分確率 $\mathrm{DP}^f_{\max}$ は，すべての非ゼロの入出力差分 $\Delta x, \Delta y$ に対する差分確率の最大値であり，

$$\mathrm{DP}^f_{\max} = \max_{\Delta x \neq 0, \Delta y} \mathrm{DP}^f(\Delta x, \Delta y)$$

と定義される．

f が暗号化関数であり，R 個の関数 $f_1, f_2, \ldots, f_R : \{0,1\}^n \to \{0,1\}^n$ の合成関数として

$$f(x) = f_R \circ f_{R-1} \circ \cdots \circ f_1(x)$$

と書ける場合を考える．各 f_i はラウンド関数に対応する．このとき，f の最大差分特性確率 $\mathrm{DCP}^f_{\max}$ は，次のように定義される．

$$\mathrm{DCP}^f_{\max} = \max_{\substack{\Delta x_0 \neq 0 \\ \Delta x_1, \ldots, \Delta x_R}} \prod_{1 \leq i \leq R} \mathrm{DP}^{f_i}(\Delta x_{i-1}, \Delta x_i)$$

ただし，f の入出力差分はそれぞれ Δx_0, Δx_R であり，f_i の入出力差分はそれぞれ Δx_{i-1}, Δx_i である．

また，このときの各ラウンド関数の入出力差分の列

$$(\Delta x_0, \Delta x_1, \ldots, \Delta x_R)$$

を差分パスという．

ある差分パスに対し，入力差分が 0 でない S-box を active S-box といい，active S-box の最小個数とは，active S-box の個数の下界を指す．

Knudsen による truncated 差分攻撃は，いくつかのビットをまとめて差分伝搬を解析する攻撃手法である [22]．最大 truncate 差分特性確率は最大差分特性確率の上界であり，最大 truncate 差分特性確率が十分小さいことは，最大差分特性確率が十分小さいことを意味する．

2.2 HyRAL に対する差分攻撃

HyRAL の差分攻撃に対する耐性評価が高木，五十嵐，金子により報告されている [34]．128 ビット鍵 HyRAL の場合，active S-box の最小個数が 37 個，192 ビット，256 ビット鍵 HyRAL の場合，active S-box の最小個数が 57 個と報告されている．S-box の最大差分確率は 2^{-6} であり，最大 truncate 差分特性確率は 128 ビット鍵 HyRAL の場合 2^{-222}，192 ビット，256 ビット鍵 HyRAL の場合 2^{-342} となる [34]．よって，128, 192, 256 ビット鍵 HyRAL はいずれも差分攻撃に対して十分な耐性を持つと考えられる．また，評価者は [34] を更新する結果を得ていない．

3　不可能差分攻撃 (128, 192, 256 ビット鍵)

3.1　不可能差分攻撃の概要

生起確率が 0 である差分パスを不可能差分特性という．不可能差分攻撃はこの不可能差分特性を生じる鍵は誤った鍵として棄却し，正しい鍵を特定する攻撃である [6].

角尾らは一般化 Feistel 構造のブロック暗号に対する不能差分特性探索法を提案した [28]．まず [28] の概要を示す．不可能差分特性は以下の 3 種類に分類される．

- 差分確率 0 を利用した特性

  差分が Δx である入力ペアを r ラウンド暗号化したとき，出力差分が Δy となることはないとき，これは r ラウンドの不可能差分特性となる．

- 中間不一致を利用した特性

  差分が Δx である入力ペアを r ラウンド暗号化したときの出力差分（の一部）が確率 1 で Δz となり，かつ，差分が Δy である出力ペアを r' ラウンド復号したときの入力差分（の一部）が確率 1 で $\Delta z'$ となり，かつ，同じビット位置での中間差分で Δz と $\Delta z'$ が $\Delta z \neq \Delta z'$ を満たすとき，$r + r'$ ラウンドの不可能差分特性となる．

- F 関数が全単射であることを利用した特性

  F 関数が全単射である場合，F 関数の入力差分が非ゼロであるにもかかわらず，出力差分が 0 となる場合，不可能差分特性となる．

また，差分を以下の Z, V, D, R の 4 種類に分類する．

- Z: Zero (差分なし)
- V: Value (特定の値の差分)
- D: Delta (任意の非ゼロの差分)
- R: Random (差分の有無不明)

上記に基づき，[28] では不能差分特性の探索手法が提案されている．

3.2　HyRAL に対する不可能差分攻撃

[28] を手法を用いた HyRAL の不可能差分攻撃に対する耐性評価が芝山，五十嵐，金子，半谷により報告されている [29, 30]．128 ビット鍵 HyRAL では 13 ラウンド，192 ビット，256 ビット鍵 HyRAL では 12 ラウンドの不可能差分特性が存在し，128 ビット鍵 HyRAL では 14 ラウンド，192 ビット鍵 HyRAL では 13 ラウンド，256 ビッ

ト鍵 HyRAL では 14 ラウンド (RK_5 がない場合は 15 ラウンド) まで不可能差分攻撃が可能であることが報告されている [29]．また，MDS 行列の分岐数を利用し，128 ビット鍵 HyRAL に対しては 13 ラウンド，192 ビット鍵，256 ビット鍵 HyRAL に対しては 12 ラウンドの不可能差分特性があることが報告されている [30]．評価者は [29, 30] を更新する結果を得ておらず，今後攻撃可能ラウンド数が伸びる可能性があるが，現在見つかっている最長の不可能差分特性と HyRAL の仕様ラウンド数には大きな開きがあり，128, 192, 256 ビット鍵 HyRAL はいずれも不可能差分攻撃に対して耐性を持つと考えられる．

4　線形攻撃 (128, 192, 256 ビット鍵)

4.1　線形攻撃の概要

松井による線形攻撃はブロック暗号に対する汎用的な攻撃法であり [25, 26]，いくつかの入出力ビット間の排他的論理和の相関関係を利用する攻撃である．[19] に沿って線形攻撃の概要を示す．

$f(x) : \{0,1\}^n \to \{0,1\}^n$ を入出力が n ビットの関数とする．入力マスク Γx と出力マスク Γy が与えられたときの f の線形確率 $\mathrm{LP}^f(\Gamma x, \Gamma y)$ は以下のように定義される．

$$\mathrm{LP}^f(\Gamma x, \Gamma y) = \left(2 \times \frac{\#\{x \in \{0,1\}^n \mid x \cdot \Gamma x = f(x) \cdot \Gamma y\}}{2^n} - 1 \right)^2$$

f の最大線形確率 $\mathrm{LP}^f_{\max}$ は，すべての非ゼロの入出力マスク $\Gamma x, \Gamma y$ に対する線形確率の最大値であり，

$$\mathrm{LP}^f_{\max} = \max_{\Gamma x, \Gamma y \neq 0} \mathrm{LP}^f(\Gamma x, \Gamma y)$$

と定義される．

f が暗号化関数であり，R 個の関数 $f_1, f_2, \ldots, f_R : \{0,1\}^n \to \{0,1\}^n$ の合成関数として

$$f(x) = f_R \circ f_{R-1} \circ \cdots \circ f_1(x)$$

と書けるとする．各 f_i はラウンド関数に対応する．このとき，f の最大線形特性確率 $\mathrm{LCP}^f_{\max}$ は，次のように定義される．

$$\mathrm{LCP}^f_{\max} = \max_{\substack{\Gamma x_0, \ldots, \Gamma x_{R-1} \\ \Gamma x_R \neq 0}} \prod_{1 \leq i \leq R} \mathrm{LP}^{f_i}(\Delta x_{i-1}, \Delta x_i)$$

ただし，f の入出力マスクはそれぞれ Γx_0, Γx_R であり，f_i の入出力マスクはそれぞれ Γx_{i-1}, Γx_i である．

また，このときの各ラウンド関数の入出力マスクの列

$$(\Gamma x_0, \Gamma x_1, \ldots, \Gamma x_R)$$

を線形パスという．

ある線形パスに対し，出力マスクが 0 でない S-box を active S-box といい，active S-box の最小個数とは，active S-box の個数の下界を指す．

Camellia の設計者らによる truncated 線形攻撃はブロック暗号に対する汎用的な攻撃である [2, 3]．最大 truncate 線形特性確率は最大線形特性確率の上界であり，最大 truncate 線形確率が十分小さいことは，最大線形特性確率が十分小さいことを意味する．

4.2 HyRAL に対する線形攻撃

HyRAL の線形攻撃に対する耐性評価が五十嵐，高木，金子により報告されている [23]．128 ビット鍵 HyRAL の場合，active S-box の最小個数が 38 個，192 ビット，256 ビット鍵 HyRAL の場合，active S-box の最小個数が 53 個と報告されている．S-box の最大線形確率は 2^{-6} であり，最大 truncate 線形特性確率は 128 ビット鍵 HyRAL の場合 2^{-228}，192 ビット，256 ビット鍵 HyRAL の場合 2^{-318} となる [23]．よって，128, 192, 256 ビット鍵 HyRAL はいずれも線形攻撃に対して十分な耐性を持つと考えられる．また，評価者は [23] を更新する結果を得ていない．

5　高階差分攻撃 (129, 130, . . . , 256 ビット鍵)

5.1　高階差分攻撃の概要

Knudsen らによる高階差分攻撃 [18, 22, 24] は，S-box などの非線形要素が次数の低いブール多項式で表現可能なブロック暗号に適用できる汎用的な攻撃法である．高階差分攻撃は，ブロック暗号のある中間ビットが d 次のブール多項式で表現できた時，$(d+1)$ 階差分をとると 0 になることを利用する．

高階差分攻撃の概要を示す．鍵 $K \in \{0,1\}^k$，平文 $M \in \{0,1\}^m$，暗号文 $C \in \{0,1\}^c$ の暗号化関数を

$$E(K, M) = C$$

とする．$(A_1, A_2, \ldots, A_d)$ を $\mathrm{GF}(2)^m$ 上の一次独立な d 個のベクトルとし，これによって張られる $\mathrm{GF}(2)^d$ 上の部分空間を $V^{(d)}$ で表す．このとき，$E(K, M)$ の M に関する d 階差分は

$$\Delta_{V^{(d)}} E(K, M) = \bigoplus_{A \in V^{(d)}} E(K, M \oplus A)$$

と定義される．

$E(K, M)$ の M に関するブール次数が d 次のとき，任意の M に対して

$$\begin{cases} \Delta_{V^{(d)}} E(K, M) = \text{constant} \\ \Delta_{V^{(d+1)}} E(K, M) = 0 \end{cases}$$

が成り立つ．

暗号化関数 E が R ラウンドからなり，R 個の関数 $f^{(i)}$ $(1 \le i \le R)$ で構成されるとき，入力 M に対する $(R-1)$ ラウンドの出力 $Y^{(R-1)}(X)$ は

$$Y^{(R-1)}(M) = f^{(R-1)}(K_{R-1}, \cdots f^{(1)}(K_1, M) \cdots)$$

と表される．ここで K_i は i ラウンドに入力するラウンド鍵である．$Y^{(R-1)}(M)$ の M に関するブール次数が d のとき，

$$\begin{cases} \Delta_{V^{(d)}} Y^{(R-1)}(M) = \text{constant} \\ \Delta_{V^{(d+1)}} Y^{(R-1)}(M) = 0 \end{cases}$$

が成り立つ．

入力 M に対する暗号文を $C(M)$ とし，$C(M)$ から $Y^{(R-1)}$ を求める関数を $f^{(-1)}$ とすると，

$$Y^{(R-1)}(M) = f^{(-1)}(K_R, C(M))$$

が得られる．したがって，

$$\begin{cases} \bigoplus_{A \in V^{(d)}} f^{(-1)}(K_R, C(M \oplus A)) = \text{constant} \\ \bigoplus_{A \in V^{(d+1)}} f^{(-1)}(K_R, C(M \oplus A)) = 0 \end{cases}$$

が成り立つ．この式を解くことにより K_R が得られる．

5.2 129, 130, . . . , 256 ビット鍵 HyRAL に対する高階差分攻撃

HyRAL の高階差分攻撃に対する耐性評価が山口，五十嵐，金子 [37]，芝山，五十嵐，金子，半谷 [31]，多賀，田中 [33] により報告されている．

128 ビット鍵 HyRAL に対し，[37] では 8 ラウンドで高階差分特性が不定になると報告されている．[31] では 128, 192, 256 ビット鍵 HyRAL に対し 7 ラウンドの飽和特性が存在し，これにより 128 ビット鍵 HyRAL では 9 ラウンドまで，192 ビット鍵 HyRAL では 10 ラウンドまで，256 ビット鍵 HyRAL では 11 ラウンドまで攻撃が可能であることが報告されている．[33] では，128 ビット鍵 HyRAL に対し 10 ラウンドまで，$129, 130, \ldots, 224$ ビット鍵 HyRAL に対し 12 ラウンドまで，$225, 226, \ldots, 256$ ビット鍵 HyRAL に対し 13 ラウンドまで攻撃可能であることが報告されている．

評価者は [32] の内容は把握しておらず，また上記のいずれをも更新する結果を得ていない．今後攻撃可能ラウンド数が伸びる可能性があるが，現在の攻撃可能ラウンド数と仕様ラウンド数とには大きな開きがあり，128, 192, 256 ビット鍵 HyRAL はいずれも高階差分攻撃に対して耐性を持つと考えられる．

6　補間攻撃 (128, 192, 256 ビット鍵)

Jakobsen と Knudsen による補間攻撃はブロック暗号に対する汎用的な攻撃法である [18]．補間攻撃の概要を示す．$y = f(x)$ を未知の多項式とする．f の次数が高々 $d-1$ であるとき，$y_i = f(x_i)$ を満たす d 個の $(x_1, y_1), \ldots, (x_d, y_d)$ から f を Lagrange の補間公式により復元できる．

このことを用いると，秘密鍵が未知であるブロック暗号の暗号化関数に対し，暗号文が平文の (係数が未知の) 多項式として記述でき，なおかつその次数が低いとき，いくつかの平文，暗号文ペアから暗号化関数を復元できる．また，次数の制限は必須ではなく，その項数が少ないだけでも十分である．

補間攻撃は S-box などの非線形関数が数学的に簡易に表現可能なブロック暗号に適用可能な攻撃法であり，補間攻撃が HyRAL の脅威になる可能性は極めて低いと考えられる．

7　代数攻撃 (128, 192, 256 ビット鍵)

XL 攻撃や [11] や XSL 攻撃 [12] などの代数攻撃では，暗号化関数の各要素の入出力を多項式で表現し，この多項式を解くことで秘密鍵を導出する．現時点ではこれらの攻撃のブロック暗号に対する有効性は十分に確立されているとはいえないと考えられる．

128 ビット鍵 HyRAL のデータ処理部には $4 \times 40 = 160$ 個の S-box が存在し，192 ビット鍵，256 ビット鍵 HyRAL のデータ処理部には $4 \times 56 = 224$ 個の S-box が存在する．AES-128 では 160 個，AES-192 では 192 個，AES-256 では 224 個の S-box が存在する．HyRAL は代数攻撃に対し AES と同等以上の耐性があると期待される．

8　等価鍵 (256 ビット鍵)

ブロック暗号 E に対し，すべての平文 M について $E_K(M) = E_{K'}(M)$ が成り立つような互いに異なる鍵のペア (K, K') を等価鍵という [20].

256 ビット鍵 HyRAL の等価鍵に関する解析を行った [4]．本章では [4] の内容をより詳細に記載する.

8.1　等価鍵の存在を示すための方針

鍵生成アルゴリズム KGA_1, KGA_2 に入力差分 ΔOK_1, ΔOK_2 を与えた時の出力差分を

$$(\Delta Y_4, \Delta Y_5, \Delta Y_6, \Delta Y_7) = \mathrm{KGA}_1(OK_1) \oplus \mathrm{KGA}_1(OK_1 \oplus \Delta OK_1) \tag{1}$$

$$(\Delta Z_4, \Delta Z_5, \Delta Z_6, \Delta Z_7) = \mathrm{KGA}_2(OK_2) \oplus \mathrm{KGA}_2(OK_2 \oplus \Delta OK_2) \tag{2}$$

とする．このとき，仮にこれらの出力差分が一致し

$$(\Delta Y_4, \Delta Y_5, \Delta Y_6, \Delta Y_7) = (\Delta Z_4, \Delta Z_5, \Delta Z_6, \Delta Z_7) \tag{3}$$

が成立すれば，鍵割り当てアルゴリズム KAA の入力 (KM_1, KM_3, KM_2, KM_4) において差分が 0 となる．この様子を図 11 に示す.

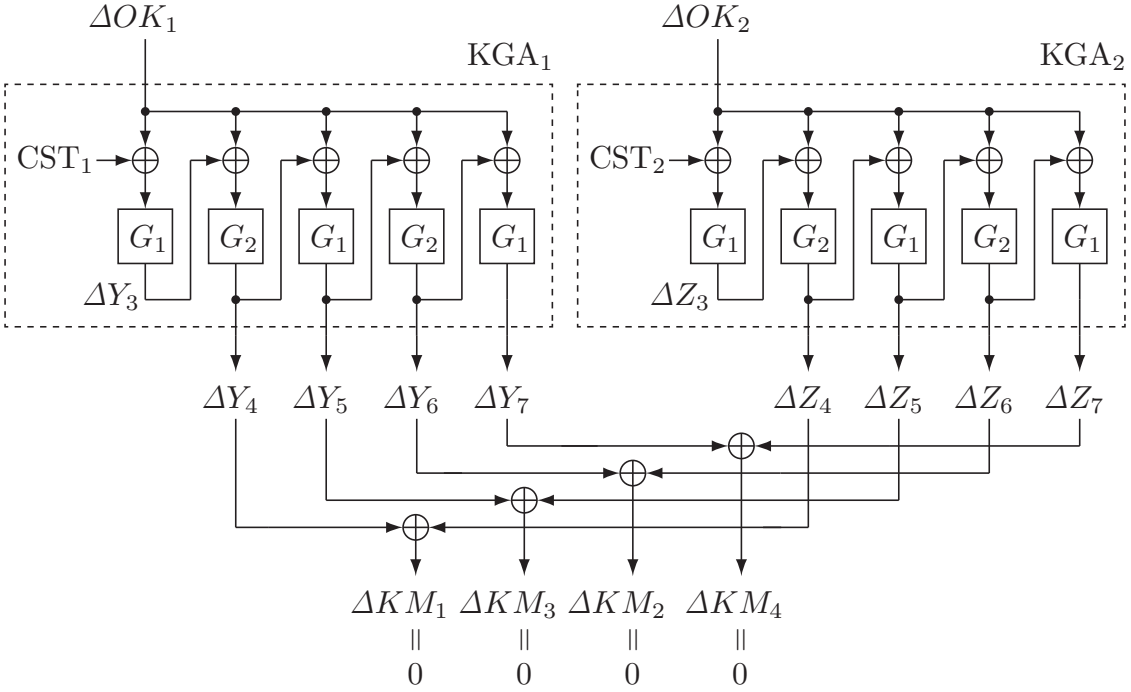

図 11: KGA_1, KGA_2 の差分の伝搬.

したがって，このとき

$$\begin{cases} (OK_1, OK_2), (OK_1 \oplus \Delta OK_1, OK_2 \oplus \Delta OK_2) \\ (OK_1 \oplus \Delta OK_1, OK_2 \oplus \Delta OK_2), (OK_1, OK_2) \end{cases} \\ \begin{cases} (OK_1 \oplus \Delta OK_1, OK_2), (OK_1, OK_2 \oplus \Delta OK_2) \\ (OK_1, OK_2 \oplus \Delta OK_2), (OK_1 \oplus \Delta OK_1, OK_2) \end{cases} \tag{4}$$

はいずれも等価鍵となる．なお，本報告書ではこれを 4 個 (2 ペア) と数える．

KGA_1, KGA_2 は定数を除いて同一のアルゴリズムであるから差分を考える上では両者を同一視でき，$\Delta OK_1 = \Delta OK_2 = \Delta OK$ として $\mathrm{KGA}_1, \mathrm{KGA}_2$ に同一の差分を入力して考える．以降，$\mathrm{KGA} \in \{\mathrm{KGA}_1, \mathrm{KGA}_2\}$ とし，KGA の差分特性を解析する．

8.2 KGA の差分特性解析 (1)

KGA を 32 ビット単位でデータを処理する関数とみなし，G_1, G_2 関数の 1 ラウンドを KGA の 1 ラウンドとみなす．このとき，KGA は 20 ラウンドからなる関数となる．

KGA の入力差分を $\Delta OK \in \{0,1\}^{128}$ と書き，出力差分を $(\Delta Y_4, \Delta Y_5, \Delta Y_6, \Delta Y_7) \in \{0,1\}^{512}$ と書く．

$r = 1, 2, \ldots, 20$ に対し，

$$\Delta X^{(r)} = (\Delta X_1^{(r)}, \Delta X_2^{(r)}, \Delta X_3^{(r)}, \Delta X_4^{(r)}) \in \{0,1\}^{128}$$

を r ラウンドの入力差分，

$$\Delta Z^{(r)} = (\Delta Z_1^{(r)}, \Delta Z_2^{(r)}, \Delta Z_3^{(r)}, \Delta Z_4^{(r)}) \in \{0,1\}^{128}$$

を出力差分とする．差分パスとは，各ラウンドの入出力差分の列

$$((\Delta X^{(1)}, \Delta Z^{(1)}), \ldots, (\Delta X^{(20)}, \Delta Z^{(20)}))$$

であり，次の条件を満たす．まず，KGA の入出力差分と対応しており，以下の式が成り立つ．

$$\begin{cases} \Delta X^{(1)} = \Delta OK \\ \Delta Z^{(8)} = \Delta Y_4 \\ \Delta Z^{(12)} = \Delta Y_5 \\ \Delta Z^{(16)} = \Delta Y_6 \\ \Delta Z^{(20)} = \Delta Y_7 \end{cases}$$

また，$r = 1, 2, \ldots, 20$ に対し

$$(\Delta X_2^{(r)}, \Delta X_3^{(r)}, \Delta X_4^{(r)}) = (\Delta Z_1^{(r)}, \Delta Z_2^{(r)}, \Delta Z_3^{(r)})$$

である．さらに，$r \in \{4, 8, 12, 16\}$ に対し

$$\Delta X^{(r+1)} = \Delta Z^{(r)} \oplus \Delta OK$$

であり，$r \in \{1, 2, \ldots, 19\} \setminus \{4, 8, 12, 16\}$ に対し

$$\Delta X^{(r+1)} = \Delta Z^{(r)}$$

である．

差分パス $((\Delta X^{(1)}, \Delta Z^{(1)}), \ldots, (\Delta X^{(20)} \Delta Z^{(20)}))$ に対する差分特性確率 $\mathrm{DCP}^{\mathrm{KGA}}$ は次式で定義される [8]．

$$\mathrm{DCP}^{\mathrm{KGA}}((\Delta X^{(1)}, \Delta Z^{(1)}), \ldots, (\Delta X^{(20)} \Delta Z^{(20)})) = \prod_{1 \le r \le 20} \mathrm{DP}^{f_i^{(r)}}(\Delta I_i^{(r)}, \Delta O_i^{(r)})$$

ただし，$f_i^{(r)}$ は r ラウンドにある f_i 関数を表す．$\Delta I_i^{(r)}$, $\Delta O_i^{(r)}$ は $f_i^{(r)}$ の入出力差分であり，

$$\begin{cases} \Delta I_i^{(r)} = \Delta X_2^{(r)} \oplus \Delta X_3^{(r)} \oplus \Delta X_4^{(r)} \\ \Delta O_i^{(r)} = \Delta X_1^{(r)} \oplus \Delta Z_4^{(r)} \end{cases}$$

である．

また，入力差分 ΔI_i，出力差分 ΔO_i に対する f_i 関数の差分確率 $\mathrm{DP}^{f_i}(\Delta I_i, \Delta O_i)$ は

$$\mathrm{DP}^{f_i}(\Delta I_i, \Delta O_i) = \frac{\#\{I \mid f_i(I) \oplus f_i(I \oplus \Delta I_i) = \Delta O_i\}}{2^{32}}$$

と定義される [8]．

ある差分パスに対し，0 でない差分が入力される f_i 関数を active f_i 関数と呼ぶ．KGA には 20 個の f_i 関数があり，active f_i 関数の個数の最大値は 20 である．

補題 1 KGA に対し，active f_i 関数の個数が 4 である差分パスが存在する．

証明 $\delta \in \{0, 1\}^{32}$ を任意の非ゼロのビット列とする．$\Delta OK = (\delta, \delta, \delta, \delta)$, $\Delta Y_4 = (\delta, \delta, 0, 0)$, $\Delta Y_5 = (0, 0, 0, \delta)$, $\Delta Y_6 = (\delta, \delta, \delta, \delta)$, $\Delta Y_7 = (0, 0, 0, 0)$ とし，表 1，及び図 12 で与えられる差分パスを考える．

表 1: 差分パスと active f_i 関数.

r	入力差分 $\Delta X^{(r)}$	出力差分 $\Delta Z^{(r)}$	active f_i 関数
1	$(\delta,\delta,\delta,\delta)$	$(\delta,\delta,\delta,0)$	f_1
2	$(\delta,\delta,\delta,0)$	$(\delta,\delta,0,\delta)$	
3	$(\delta,\delta,0,\delta)$	$(\delta,0,\delta,\delta)$	
4	$(\delta,0,\delta,\delta)$	$(0,\delta,\delta,\delta)$	
5	$(\delta,0,0,0)$	$(0,0,0,\delta)$	
6	$(0,0,0,\delta)$	$(0,0,\delta,\delta)$	f_7
7	$(0,0,\delta,\delta)$	$(0,\delta,\delta,0)$	
8	$(0,\delta,\delta,0)$	$(\delta,\delta,0,0)$	
9	$(0,0,\delta,\delta)$	$(0,\delta,\delta,0)$	
10	$(0,\delta,\delta,0)$	$(\delta,\delta,0,0)$	
11	$(\delta,\delta,0,0)$	$(\delta,0,0,0)$	f_3
12	$(\delta,0,0,0)$	$(0,0,0,\delta)$	
13	$(\delta,\delta,\delta,0)$	$(\delta,\delta,0,\delta)$	
14	$(\delta,\delta,0,\delta)$	$(\delta,0,\delta,\delta)$	
15	$(\delta,0,\delta,\delta)$	$(0,\delta,\delta,\delta)$	
16	$(0,\delta,\delta,\delta)$	$(\delta,\delta,\delta,\delta)$	f_5
17	$(0,0,0,0)$	$(0,0,0,0)$	
18	$(0,0,0,0)$	$(0,0,0,0)$	
19	$(0,0,0,0)$	$(0,0,0,0)$	
20	$(0,0,0,0)$	$(0,0,0,0)$	

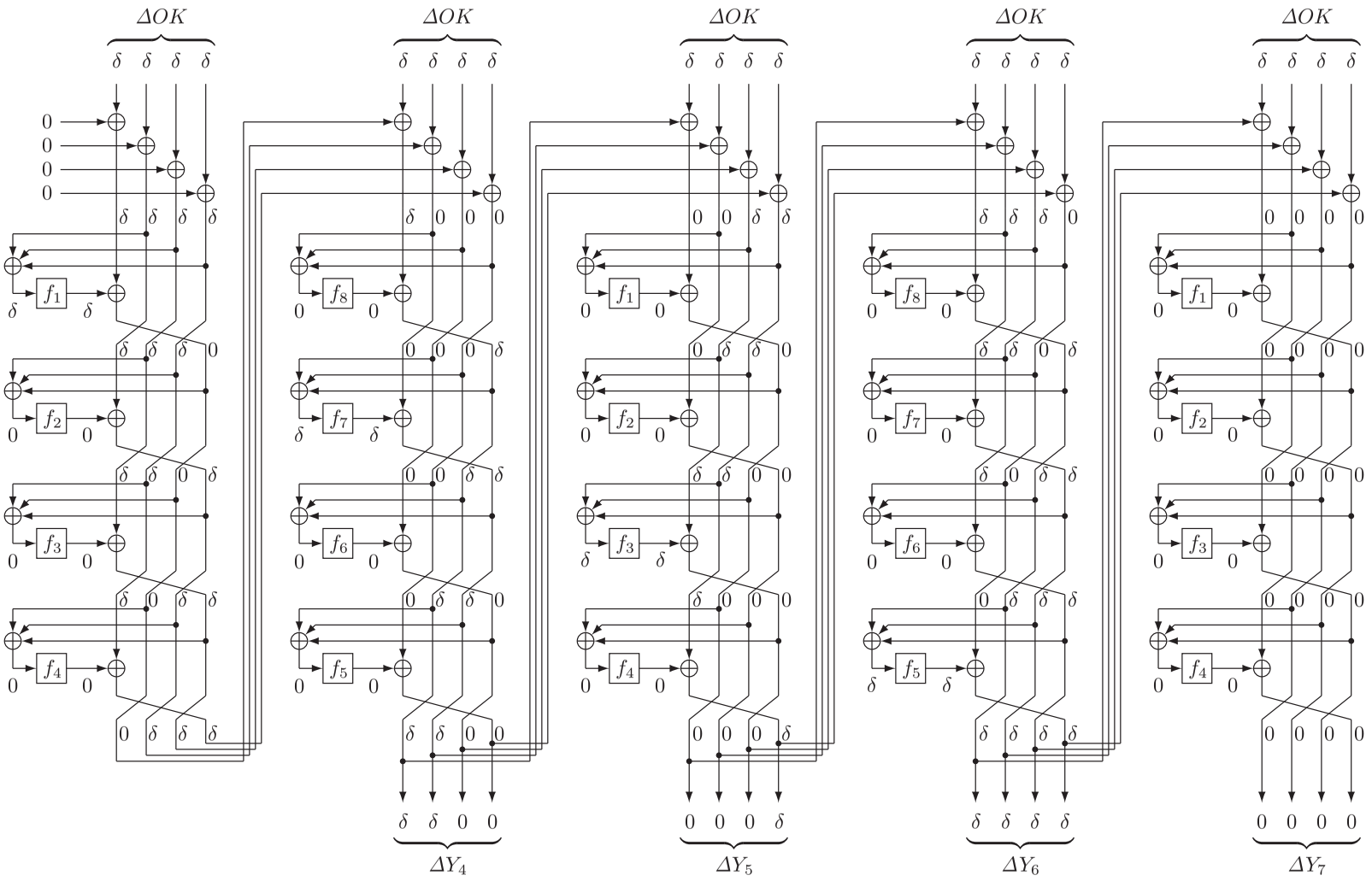

図 12: active f_i 関数の個数が 4 である差分パス.

この差分パスに対する active f_i 関数は $f_1^{(1)}, f_7^{(6)}, f_3^{(11)}, f_5^{(16)}$ の 4 個である．　□

f_i 関数の入出力差分がともに δ となるという条件のもとで，KGA に対する 15 通りの入力差分

$$(0,0,0,\delta),(0,0,\delta,0),\ldots,(\delta,\delta,\delta,\delta)$$

の active f_i 関数の個数を求めた．その結果を表 2 にまとめる．

表 2: 各入力差分に対する active f_i 関数の個数．

入力差分 ΔOK	個数
$(0,0,0,\delta)$	9
$(0,0,\delta,0)$	9
$(0,0,\delta,\delta)$	10
$(0,\delta,0,0)$	9
$(0,\delta,0,\delta)$	10
$(0,\delta,\delta,0)$	10
$(0,\delta,\delta,\delta)$	7
$(\delta,0,0,0)$	9
$(\delta,0,0,\delta)$	10
$(\delta,0,\delta,0)$	10
$(\delta,0,\delta,\delta)$	7
$(\delta,\delta,0,0)$	10
$(\delta,\delta,0,\delta)$	7
$(\delta,\delta,\delta,0)$	7
$(\delta,\delta,\delta,\delta)$	4

表 2 より，この条件のもとでは補題 1 の差分パスに対する active f_i 関数の個数が最少である．

8.3 KGA の差分特性解析 (2)

f_i 関数に対し，入出力差分がともに δ となる確率を

$$\mathrm{DP}^{f_i}(\delta) = \mathrm{DP}^{f_i}(\delta,\delta) = \frac{\#\{I \mid f_i(I) \oplus f_i(I \oplus \delta) = \delta\}}{2^{32}}$$

と書く．

補題 1 の差分パスに対する差分特性確率は δ にのみ依存し，これを $\mathrm{DCP}^{\mathrm{KGA}}(\delta)$ と書く．これは次式で与えられる．

$$\mathrm{DCP}^{\mathrm{KGA}}(\delta) = \mathrm{DP}^{f_1}(\delta) \times \mathrm{DP}^{f_3}(\delta) \times \mathrm{DP}^{f_5}(\delta) \times \mathrm{DP}^{f_7}(\delta)$$

$\mathrm{DCP}^{\mathrm{KGA}}(\delta)$ について，次の補題を示す．

補題 2 $\mathrm{DCP}^{\mathrm{KGA}}(\delta) > 2^{-128}$ となる δ が存在する．

証明 `0x00000000`, . . . , `0xffffffff` の 2^{32} 通りの δ に対し，$\mathrm{DCP}^{\mathrm{KGA}}(\delta)$ の値を求めた．

これは以下の手順で求められる．

1. S-box の差分分布表 $S(\Delta x, \Delta y)$ を作成する．$\Delta x, \Delta y \in \{$`0x00`, `0x01`, . . . , `0xff`$\}$ であり，
$$S(\Delta x, \Delta y) = \#\{x \mid S(x) \oplus S(x \oplus \Delta x) = \Delta y\}$$
である．

2. $i \in \{1, 3, 5, 7\}$ と $\delta \in \{$`0x00000000`, . . . , `0xffffffff`$\}$ を任意に固定する．

3. $(\Delta x_1, \Delta x_2, \Delta x_3, \Delta x_4) \leftarrow T_i(\delta)$ とする．

4. $\delta = (\delta_1, \delta_2, \delta_3, \delta_4)$ とバイトに分割し，
$$\begin{pmatrix} \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \\ \Delta y_4 \end{pmatrix} = \begin{pmatrix} \texttt{0x03} & \texttt{0x03} & \texttt{0x02} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x02} & \texttt{0x02} \\ \texttt{0x07} & \texttt{0x03} & \texttt{0x01} & \texttt{0x02} \\ \texttt{0x07} & \texttt{0x04} & \texttt{0x05} & \texttt{0x03} \end{pmatrix}^{-1} \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{pmatrix}$$
とする．

5. $\mathrm{DP}^{f_i}(\delta) = \dfrac{S(\Delta x_1, \Delta y_1) \cdot S(\Delta x_2, \Delta y_2) \cdot S(\Delta x_3, \Delta y_3) \cdot S(\Delta x_4, \Delta y_4)}{2^{32}}$ である．

上記の手順により $\mathrm{DP}^{f_1}(\delta)$, $\mathrm{DP}^{f_3}(\delta)$, $\mathrm{DP}^{f_5}(\delta)$, $\mathrm{DP}^{f_7}(\delta)$ をそれぞれ求め，それらの積が $\mathrm{DCP}^{\mathrm{KGA}}(\delta)$ である．

この結果を表 3 にまとめる．

表 3 より，$\mathrm{DCP}^{\mathrm{KGA}}(\delta) > 2^{-128}$ となる δ が 89938 通り存在する． □

なお，$\delta =$ `0xd7d7d0d7` に対しては

$$\begin{cases} \mathrm{DP}^{f_1}(\delta) = 2^{-25} \\ \mathrm{DP}^{f_3}(\delta) = 2^{-26} \\ \mathrm{DP}^{f_5}(\delta) = 2^{-26} \\ \mathrm{DP}^{f_7}(\delta) = 2^{-26} \end{cases} \tag{5}$$

である．

表 3: $\mathrm{DCP}^{\mathrm{KGA}}(\delta) > 2^{-128}$ となる δ の例と個数.

$\mathrm{DCP}^{\mathrm{KGA}}(\delta)$	δ の例	δ の個数
2^{-103}	0xd7d7d0d7	1
2^{-104}	0xc5c5d254	1
2^{-105}	0x4e4ec554	1
2^{-106}	0x3c3cf4ff	8
2^{-107}	0x6161f9d9	1
2^{-108}	0x054d9797	34
2^{-109}	0x0101019a	157
2^{-110}	0x0159591a	1579
2^{-111}	0x0101e818	7685
2^{-112}	0x01010520	80471

8.4 等価鍵の存在性

$\mathrm{DCP}^{\mathrm{KGA}}(\delta) > 2^{-128}$ となる δ を考える.
OK_1 をランダムに選択すると確率 $\mathrm{DCP}^{\mathrm{KGA}}(\delta)$ で

$$\begin{cases} \Delta OK_1 = (\delta, \delta, \delta, \delta) \\ \Delta Y_4 = (\delta, \delta, 0, 0) \\ \Delta Y_5 = (0, 0, 0, \delta) \\ \Delta Y_6 = (\delta, \delta, \delta, \delta) \\ \Delta Y_7 = (0, 0, 0, 0) \end{cases} \tag{6}$$

に対し，式 (1) が成立する．これは，少なくとも $2^{128} \times \mathrm{DCP}^{\mathrm{KGA}}(\delta)$ 通りの OK_1 が式 (1) を満たすことを意味する．同様に，

$$\begin{cases} \Delta OK_2 = (\delta, \delta, \delta, \delta) \\ \Delta Z_4 = (\delta, \delta, 0, 0) \\ \Delta Z_5 = (0, 0, 0, \delta) \\ \Delta Z_6 = (\delta, \delta, \delta, \delta) \\ \Delta Z_7 = (0, 0, 0, 0) \end{cases} \tag{7}$$

に対し，$2^{128} \times \mathrm{DCP}^{\mathrm{KGA}}(\delta)$ 通りの OK_2 が式 (2) を満たす．式 (1)，式 (6)，式 (2)，式 (7) を同時に満たすような (OK_1, OK_2) は，式 (3) も満たし，式 (4) にある 4 個 (2 ペア) の等価鍵の存在を意味する．重複を考慮し，表 3 より等価鍵の個数は以下のように計算できる.

$$\frac{4 \times (2^{50} \times 1 + 2^{48} \times 1 + \cdots + 2^{32} \times 80471)}{4} \geq 2^{51.0}$$

また，ペア数はこれの 1/2 である $2^{50.0}$ となる．

以上より，次の補題を得る．

補題 3 256 ビット鍵 HyRAL には $2^{51.0}$ 個 ($2^{50.0}$ ペア) の等価鍵が存在する．

8.5 等価鍵導出アルゴリズム

本章では，等価鍵を導出するアルゴリズムを提案する．

前章と同様に，$\mathrm{KGA} \in \{\mathrm{KGA}_1, \mathrm{KGA}_2\}$ を 32 ビット単位でデータを処理する関数とみなし，G_1, G_2 関数の 1 ラウンドを KGA の 1 ラウンドとする．KGA の入力を $OK \in \{OK_1, OK_2\}$ とし，$OK = (K_1, K_2, K_3, K_4) \in \{0,1\}^{128}$ と書く．また，$\mathrm{CST} \in \{\mathrm{CST}_1, \mathrm{CST}_2\}$ とし，$\mathrm{CST} = (C_1, C_2, C_3, C_4) \in \{0,1\}^{128}$ と書く．KGA は 20 ラウンドの関数であり，$r = 1, 2, \ldots, 20$ に対し，r ラウンドにある f_i 関数 $f_i^{(r)}$ の入出力をそれぞれ $I_i^{(r)}, O_i^{(r)} \in \{0,1\}^{32}$ と書く．図 13 に KGA のはじめの 8 ラウンドを示す．

$\delta =$ `0xd7d7d0d7` の場合を考える．$i \in \{1,3,5,7\}$ に対し，$f_i(I_i) \oplus f_i(I_i \oplus \delta) = \delta$ を満たす I_i のリストを $\mathcal{I}_i$ とする．式 (5) より，それぞれのサイズは 128, 64, 64, 64 となり，これらのリストを

$$\mathcal{I}_1 = \{I_1[0], \ldots, I_1[127]\}$$
$$\mathcal{I}_3 = \{I_3[0], \ldots, I_3[63]\}$$
$$\mathcal{I}_5 = \{I_5[0], \ldots, I_5[63]\}$$
$$\mathcal{I}_7 = \{I_7[0], \ldots, I_7[63]\}$$

とする．実際に求めた $\mathcal{I}_1, \mathcal{I}_3, \mathcal{I}_5, \mathcal{I}_7$ を次に示す．

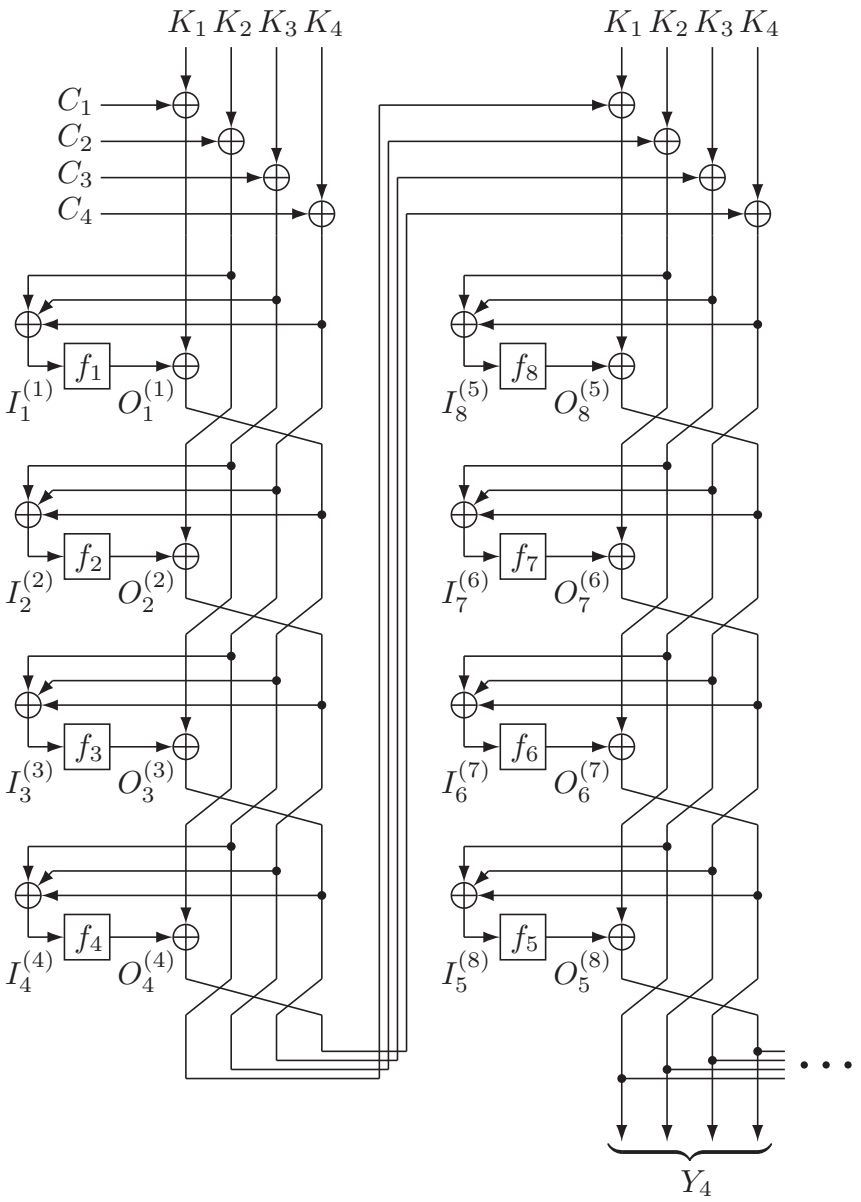

図 13: KGA のはじめの 8 ラウンド.

$$\mathcal{I}_1 = \{\texttt{0x17170c17}, \texttt{0x17170c53}, \texttt{0x17170c84}, \texttt{0x17170cc0},$$
$$\texttt{0x1717dc17}, \texttt{0x1717dc53}, \texttt{0x1717dc84}, \texttt{0x1717dcc0},$$
$$\texttt{0x17530c17}, \texttt{0x17530c53}, \texttt{0x17530c84}, \texttt{0x17530cc0},$$
$$\texttt{0x1753dc17}, \texttt{0x1753dc53}, \texttt{0x1753dc84}, \texttt{0x1753dcc0},$$
$$\texttt{0x17840c17}, \texttt{0x17840c53}, \texttt{0x17840c84}, \texttt{0x17840cc0},$$
$$\texttt{0x1784dc17}, \texttt{0x1784dc53}, \texttt{0x1784dc84}, \texttt{0x1784dcc0},$$
$$\texttt{0x17c00c17}, \texttt{0x17c00c53}, \texttt{0x17c00c84}, \texttt{0x17c00cc0},$$
$$\texttt{0x17c0dc17}, \texttt{0x17c0dc53}, \texttt{0x17c0dc84}, \texttt{0x17c0dcc0},$$
$$\texttt{0x53170c17}, \texttt{0x53170c53}, \texttt{0x53170c84}, \texttt{0x53170cc0},$$
$$\texttt{0x5317dc17}, \texttt{0x5317dc53}, \texttt{0x5317dc84}, \texttt{0x5317dcc0},$$
$$\texttt{0x53530c17}, \texttt{0x53530c53}, \texttt{0x53530c84}, \texttt{0x53530cc0},$$
$$\texttt{0x5353dc17}, \texttt{0x5353dc53}, \texttt{0x5353dc84}, \texttt{0x5353dcc0},$$
$$\texttt{0x53840c17}, \texttt{0x53840c53}, \texttt{0x53840c84}, \texttt{0x53840cc0},$$
$$\texttt{0x5384dc17}, \texttt{0x5384dc53}, \texttt{0x5384dc84}, \texttt{0x5384dcc0},$$
$$\texttt{0x53c00c17}, \texttt{0x53c00c53}, \texttt{0x53c00c84}, \texttt{0x53c00cc0},$$
$$\texttt{0x53c0dc17}, \texttt{0x53c0dc53}, \texttt{0x53c0dc84}, \texttt{0x53c0dcc0},$$
$$\texttt{0x84170c17}, \texttt{0x84170c53}, \texttt{0x84170c84}, \texttt{0x84170cc0},$$
$$\texttt{0x8417dc17}, \texttt{0x8417dc53}, \texttt{0x8417dc84}, \texttt{0x8417dcc0},$$
$$\texttt{0x84530c17}, \texttt{0x84530c53}, \texttt{0x84530c84}, \texttt{0x84530cc0},$$
$$\texttt{0x8453dc17}, \texttt{0x8453dc53}, \texttt{0x8453dc84}, \texttt{0x8453dcc0},$$
$$\texttt{0x84840c17}, \texttt{0x84840c53}, \texttt{0x84840c84}, \texttt{0x84840cc0},$$
$$\texttt{0x8484dc17}, \texttt{0x8484dc53}, \texttt{0x8484dc84}, \texttt{0x8484dcc0},$$
$$\texttt{0x84c00c17}, \texttt{0x84c00c53}, \texttt{0x84c00c84}, \texttt{0x84c00cc0},$$
$$\texttt{0x84c0dc17}, \texttt{0x84c0dc53}, \texttt{0x84c0dc84}, \texttt{0x84c0dcc0},$$
$$\texttt{0xc0170c17}, \texttt{0xc0170c53}, \texttt{0xc0170c84}, \texttt{0xc0170cc0},$$
$$\texttt{0xc017dc17}, \texttt{0xc017dc53}, \texttt{0xc017dc84}, \texttt{0xc017dcc0},$$
$$\texttt{0xc0530c17}, \texttt{0xc0530c53}, \texttt{0xc0530c84}, \texttt{0xc0530cc0},$$
$$\texttt{0xc053dc17}, \texttt{0xc053dc53}, \texttt{0xc053dc84}, \texttt{0xc053dcc0},$$
$$\texttt{0xc0840c17}, \texttt{0xc0840c53}, \texttt{0xc0840c84}, \texttt{0xc0840cc0},$$
$$\texttt{0xc084dc17}, \texttt{0xc084dc53}, \texttt{0xc084dc84}, \texttt{0xc084dcc0},$$
$$\texttt{0xc0c00c17}, \texttt{0xc0c00c53}, \texttt{0xc0c00c84}, \texttt{0xc0c00cc0},$$
$$\texttt{0xc0c0dc17}, \texttt{0xc0c0dc53}, \texttt{0xc0c0dc84}, \texttt{0xc0c0dcc0}\}$$

$$\mathcal{I}_3 = \{\texttt{0x2b172917}, \texttt{0x2b172953}, \texttt{0x2b172984}, \texttt{0x2b1729c0},$$
$$\texttt{0x2b17f917}, \texttt{0x2b17f953}, \texttt{0x2b17f984}, \texttt{0x2b17f9c0},$$
$$\texttt{0x2b532917}, \texttt{0x2b532953}, \texttt{0x2b532984}, \texttt{0x2b5329c0},$$
$$\texttt{0x2b53f917}, \texttt{0x2b53f953}, \texttt{0x2b53f984}, \texttt{0x2b53f9c0},$$
$$\texttt{0x2b842917}, \texttt{0x2b842953}, \texttt{0x2b842984}, \texttt{0x2b8429c0},$$
$$\texttt{0x2b84f917}, \texttt{0x2b84f953}, \texttt{0x2b84f984}, \texttt{0x2b84f9c0},$$
$$\texttt{0x2bc02917}, \texttt{0x2bc02953}, \texttt{0x2bc02984}, \texttt{0x2bc029c0},$$
$$\texttt{0x2bc0f917}, \texttt{0x2bc0f953}, \texttt{0x2bc0f984}, \texttt{0x2bc0f9c0},$$
$$\texttt{0xfc172917}, \texttt{0xfc172953}, \texttt{0xfc172984}, \texttt{0xfc1729c0},$$
$$\texttt{0xfc17f917}, \texttt{0xfc17f953}, \texttt{0xfc17f984}, \texttt{0xfc17f9c0},$$
$$\texttt{0xfc532917}, \texttt{0xfc532953}, \texttt{0xfc532984}, \texttt{0xfc5329c0},$$
$$\texttt{0xfc53f917}, \texttt{0xfc53f953}, \texttt{0xfc53f984}, \texttt{0xfc53f9c0},$$
$$\texttt{0xfc842917}, \texttt{0xfc842953}, \texttt{0xfc842984}, \texttt{0xfc8429c0},$$
$$\texttt{0xfc84f917}, \texttt{0xfc84f953}, \texttt{0xfc84f984}, \texttt{0xfc84f9c0},$$
$$\texttt{0xfcc02917}, \texttt{0xfcc02953}, \texttt{0xfcc02984}, \texttt{0xfcc029c0},$$
$$\texttt{0xfcc0f917}, \texttt{0xfcc0f953}, \texttt{0xfcc0f984}, \texttt{0xfcc0f9c0}\}$$

$$\mathcal{I}_5 = \{\texttt{0x172b2917}, \texttt{0x172b2953}, \texttt{0x172b2984}, \texttt{0x172b29c0},$$
$$\texttt{0x172bf917}, \texttt{0x172bf953}, \texttt{0x172bf984}, \texttt{0x172bf9c0},$$
$$\texttt{0x17fc2917}, \texttt{0x17fc2953}, \texttt{0x17fc2984}, \texttt{0x17fc29c0},$$
$$\texttt{0x17fcf917}, \texttt{0x17fcf953}, \texttt{0x17fcf984}, \texttt{0x17fcf9c0},$$
$$\texttt{0x532b2917}, \texttt{0x532b2953}, \texttt{0x532b2984}, \texttt{0x532b29c0},$$
$$\texttt{0x532bf917}, \texttt{0x532bf953}, \texttt{0x532bf984}, \texttt{0x532bf9c0},$$
$$\texttt{0x53fc2917}, \texttt{0x53fc2953}, \texttt{0x53fc2984}, \texttt{0x53fc29c0},$$
$$\texttt{0x53fcf917}, \texttt{0x53fcf953}, \texttt{0x53fcf984}, \texttt{0x53fcf9c0},$$
$$\texttt{0x842b2917}, \texttt{0x842b2953}, \texttt{0x842b2984}, \texttt{0x842b29c0},$$
$$\texttt{0x842bf917}, \texttt{0x842bf953}, \texttt{0x842bf984}, \texttt{0x842bf9c0},$$
$$\texttt{0x84fc2917}, \texttt{0x84fc2953}, \texttt{0x84fc2984}, \texttt{0x84fc29c0},$$
$$\texttt{0x84fcf917}, \texttt{0x84fcf953}, \texttt{0x84fcf984}, \texttt{0x84fcf9c0},$$
$$\texttt{0xc02b2917}, \texttt{0xc02b2953}, \texttt{0xc02b2984}, \texttt{0xc02b29c0},$$
$$\texttt{0xc02bf917}, \texttt{0xc02bf953}, \texttt{0xc02bf984}, \texttt{0xc02bf9c0},$$
$$\texttt{0xc0fc2917}, \texttt{0xc0fc2953}, \texttt{0xc0fc2984}, \texttt{0xc0fc29c0},$$
$$\texttt{0xc0fcf917}, \texttt{0xc0fcf953}, \texttt{0xc0fcf984}, \texttt{0xc0fcf9c0}\}$$

$$\begin{aligned}\mathcal{I}_7 = \{&\texttt{0x1717292b}, \texttt{0x171729fc}, \texttt{0x1717f92b}, \texttt{0x1717f9fc},\\ &\texttt{0x1753292b}, \texttt{0x175329fc}, \texttt{0x1753f92b}, \texttt{0x1753f9fc},\\ &\texttt{0x1784292b}, \texttt{0x178429fc}, \texttt{0x1784f92b}, \texttt{0x1784f9fc},\\ &\texttt{0x17c0292b}, \texttt{0x17c029fc}, \texttt{0x17c0f92b}, \texttt{0x17c0f9fc},\\ &\texttt{0x5317292b}, \texttt{0x531729fc}, \texttt{0x5317f92b}, \texttt{0x5317f9fc},\\ &\texttt{0x5353292b}, \texttt{0x535329fc}, \texttt{0x5353f92b}, \texttt{0x5353f9fc},\\ &\texttt{0x5384292b}, \texttt{0x538429fc}, \texttt{0x5384f92b}, \texttt{0x5384f9fc},\\ &\texttt{0x53c0292b}, \texttt{0x53c029fc}, \texttt{0x53c0f92b}, \texttt{0x53c0f9fc},\\ &\texttt{0x8417292b}, \texttt{0x841729fc}, \texttt{0x8417f92b}, \texttt{0x8417f9fc},\\ &\texttt{0x8453292b}, \texttt{0x845329fc}, \texttt{0x8453f92b}, \texttt{0x8453f9fc},\\ &\texttt{0x8484292b}, \texttt{0x848429fc}, \texttt{0x8484f92b}, \texttt{0x8484f9fc},\\ &\texttt{0x84c0292b}, \texttt{0x84c029fc}, \texttt{0x84c0f92b}, \texttt{0x84c0f9fc},\\ &\texttt{0xc017292b}, \texttt{0xc01729fc}, \texttt{0xc017f92b}, \texttt{0xc017f9fc},\\ &\texttt{0xc053292b}, \texttt{0xc05329fc}, \texttt{0xc053f92b}, \texttt{0xc053f9fc},\\ &\texttt{0xc084292b}, \texttt{0xc08429fc}, \texttt{0xc084f92b}, \texttt{0xc084f9fc},\\ &\texttt{0xc0c0292b}, \texttt{0xc0c029fc}, \texttt{0xc0c0f92b}, \texttt{0xc0c0f9fc}\}\end{aligned}$$

ここで,

$$I_1^{(1)} \in \mathcal{I}_1, I_7^{(6)} \in \mathcal{I}_7, I_3^{(11)} \in \mathcal{I}_3, I_5^{(16)} \in \mathcal{I}_5$$

となる (K_1, K_2, K_3, K_4) を導出できれば，それは式 (1) 及び式 (6)，あるいは式 (2) 及び式 (7) を満たす OK を導出できたことを意味する．

$I_1^{(1)} \in \mathcal{I}_1$ を満たす (K_1, K_2, K_3, K_4) を導出することは容易であり，これは

$$K_2 \oplus C_2 \oplus K_3 \oplus C_3 \oplus K_4 \oplus C_4 \in \mathcal{I}_1$$

を満たす (K_1, K_2, K_3, K_4) である．これに対し，$I_1^{(1)} \in \mathcal{I}_1$, $I_7^{(6)} \in \mathcal{I}_7$ を両方満たす (K_1, K_2, K_3, K_4) を導出できることを示す．

補題 4 任意に固定された $\tilde{K}_1$, $I_1^{(1)}$, $I_8^{(5)}$, $I_7^{(6)}$ に対する (K_1, K_2, K_3, K_4) を導出できる．ただし，$\tilde{K}_1 = K_1 \oplus K_3$ である．

証明 $I_1^{(1)}$, $I_8^{(5)}$ が固定されているので，$O_1^{(1)} = f_1(I_1^{(1)})$, $O_8^{(5)} = f_8(I_8^{(5)})$ も固定されている．まず $\tilde{C}_1, \ldots, \tilde{C}_5$ を以下のように置く．

$$\begin{cases} \tilde{C}_1 = C_1 \oplus C_3 \oplus C_4 \oplus O_1^{(1)} \\ \tilde{C}_2 = C_1 \oplus C_3 \oplus I_1^{(1)} \oplus O_1^{(1)} \\ \tilde{C}_3 = C_1 \oplus C_4 \oplus I_1^{(1)} \oplus O_1^{(1)} \\ \tilde{C}_4 = C_2 \oplus C_3 \oplus C_4 \\ \tilde{C}_5 = C_1 \oplus C_2 \oplus O_1^{(1)} \oplus I_7^{(6)} \end{cases}$$

これらはいずれも固定された定数である．また，$\tilde{K_2} = K_1 \oplus K_3 \oplus K_4$, $\tilde{K_3} = K_1 \oplus K_4$ と置く．

まず $I_1^{(1)}$ について $I_1^{(1)} = K_2 \oplus C_2 \oplus K_3 \oplus C_3 \oplus K_4 \oplus C_4$ が成り立つ必要があり，これは

$$K_2 = I_1^{(1)} \oplus C_2 \oplus K_3 \oplus C_3 \oplus K_4 \oplus C_4 \tag{8}$$

と等価である．

次に，$I_2^{(2)} = \tilde{K}_2 \oplus \tilde{C}_1$ であるから

$$O_2^{(2)} = f_2(\tilde{K}_2 \oplus \tilde{C}_1) \tag{9}$$

と書ける．また，式 (8) を用いると $I_3^{(3)} = \tilde{K}_1 \oplus \tilde{C}_2 \oplus O_2^{(2)}$ であるから

$$O_3^{(3)} = f_3(\tilde{K}_1 \oplus \tilde{C}_2 \oplus O_2^{(2)}) \tag{10}$$

と書ける．さらに，式 (8) より $I_4^{(4)} = \tilde{K}_3 \oplus \tilde{C}_3 \oplus O_2^{(2)} \oplus O_3^{(3)}$ であるから

$$O_4^{(4)} = f_4(\tilde{K}_3 \oplus \tilde{C}_3 \oplus O_2^{(2)} \oplus O_3^{(3)}) \tag{11}$$

と書ける．

ここで 5 ラウンドの入力は

$$(C_1 \oplus O_1^{(1)}, C_2 \oplus O_2^{(2)}, C_3 \oplus O_3^{(3)}, C_4 \oplus O_4^{(4)})$$

であるから，

$$I_8^{(5)} = \tilde{C}_4 \oplus O_2^{(2)} \oplus O_3^{(3)} \oplus O_4^{(4)} \tag{12}$$

が満たされなければならない．また，

$$I_7^{(6)} = C_1 \oplus C_3 \oplus C_4 \oplus O_1^{(1)} \oplus O_3^{(3)} \oplus O_4^{(4)} \oplus O_8^{(5)}$$

も満たされる必要があり，式 (12) よりこれは

$$\tilde{C}_5 \oplus O_2^{(2)} \oplus I_8^{(5)} = O_8^{(5)} \tag{13}$$

と等価である．

$\tilde{C}_5$, $I_8^{(5)}$, $O_8^{(5)}$ は固定されているので式 (13) を満たす $O_2^{(2)}$ が定まる．$O_2^{(2)}$ が定まると式 (9) を満たす $\tilde{K}_2$ が

$$\tilde{K}_2 = f_2^{-1}(O_2^{(2)}) \oplus \tilde{C}_1$$

と求められる．また，$O_2^{(2)}$ が定まり $\tilde{K}_1$ が固定されているので式 (10) を満たす $O_3^{(3)}$ が定まる．$O_2^{(2)}$, $O_3^{(3)}$ がともに定まると式 (12) を満たす $O_4^{(4)}$ が定まり，これらの $O_2^{(2)}$, $O_3^{(3)}$, $O_4^{(4)}$ に対し，式 (11) より $\tilde{K}_3$ を

$$\tilde{K}_3 = f_4^{-1}(O_4^{(4)}) \oplus \tilde{C}_3 \oplus O_2^{(2)} \oplus O_3^{(3)}$$

と求めることができる．このときの (K_1, K_2, K_3, K_4) は $(K_1, K_2, K_3, K_4) \leftarrow (\tilde{K}_1 \oplus \tilde{K}_2 \oplus \tilde{K}_3, \tilde{K}_1 \oplus \tilde{K}_3 \oplus I_1^{(1)} \oplus \tilde{C}_4, \tilde{K}_2 \oplus \tilde{K}_3, \tilde{K}_1 \oplus \tilde{K}_2)$ である． □

補題 4 を用いて等価鍵導出アルゴリズムを提案する．

1. $I_1^{(1)} \in \mathcal{I}_1$, $I_7^{(6)} \in \mathcal{I}_7$ を満たす $I_1^{(1)}$, $I_7^{(6)}$ を任意に固定する．
2. $I_8^{(5)}, \tilde{K}_1$ を任意に固定する．
3. 補題 4 により (K_1, K_2, K_3, K_4) を求める．
4. (K_1, K_2, K_3, K_4) より $I_3^{(11)}$ を計算し，$I_3^{(11)} \in \mathcal{I}_3$ が成り立てばステップ 5 へ進む．そうでなければステップ 2 へ戻る．
5. (K_1, K_2, K_3, K_4) より $I_5^{(16)}$ を計算し，$I_5^{(16)} \in \mathcal{I}_5$ が成り立てば (K_1, K_2, K_3, K_4) を出力して終了する．そうでなければステップ 2 へ戻る．

$I_3^{(11)}$, $I_5^{(16)}$ が $\{0,1\}^{32}$ 上を一様分布すると仮定すると，$I_3^{(11)} \in \mathcal{I}_3$, $I_5^{(16)} \in \mathcal{I}_5$ が両方成立する確率は $(64/2^{32})^2 = 2^{-52}$ であるから，2^{52} 通りの $(I_8^{(5)}, \tilde{K}_1)$ を探索すればステップ 5 において (K_1, K_2, K_3, K_4) が出力される．

8.6 アルゴリズムの計算量

前章で提案した等価鍵導出アルゴリズムでは，2^{52} 通りの $(I_8^{(5)}, \tilde{K}_1)$ に対し，$I_3^{(11)} \in \mathcal{I}_3$ のチェックを行う．この $I_3^{(11)}$ のチェックが主要な計算量であり，アルゴリズムの実装においては次の補題を用いる．

補題 5 任意に固定された $\tilde{K}_1$, $I_1^{(1)}$, $O_1^{(1)}$, $I_8^{(5)}$, $I_7^{(6)}$, $O_7^{(6)}$ に対し，f_i 関数を 7 回計算することで $I_3^{(11)}$ を計算できる．

証明 次の手順で計算できる．

1. $O_8^{(5)} \leftarrow f_8(I_8^{(5)})$
2. $O_2^{(2)} \leftarrow \tilde{C}_5 \oplus I_8^{(5)} \oplus O_8^{(5)}$
3. $\tilde{K}_2 \leftarrow f_2^{-1}(O_2^{(2)}) \oplus \tilde{C}_1$
4. $O_3^{(3)} \leftarrow f_3(\tilde{K}_1 \oplus \tilde{C}_2 \oplus O_2^{(2)})$
5. $O_4^{(4)} \leftarrow \tilde{C}_4 \oplus I_8^{(5)} \oplus O_2^{(2)} \oplus O_3^{(3)}$
6. $O_6^{(7)} \leftarrow f_6(C_1 \oplus C_2 \oplus C_4 \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_7^{(6)})$
7. $O_5^{(8)} \leftarrow f_5(C_1 \oplus C_2 \oplus C_3 \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_3^{(3)} \oplus O_8^{(5)} \oplus O_7^{(6)} \oplus O_6^{(7)})$
8. $O_1^{(9)} \leftarrow f_1(I_1^{(1)} \oplus O_2^{(2)} \oplus O_3^{(3)} \oplus O_4^{(4)} \oplus O_7^{(6)} \oplus O_6^{(7)} \oplus O_5^{(8)})$
9. $O_2^{(10)} \leftarrow f_2(\tilde{K}_2 \oplus \tilde{C}_1 \oplus O_3^{(3)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_6^{(7)} \oplus O_5^{(8)} \oplus O_1^{(9)})$
10. $I_3^{(11)} \leftarrow \tilde{K}_1 \oplus C_1 \oplus C_3 \oplus I_1^{(1)} \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_7^{(6)} \oplus O_5^{(8)} \oplus O_1^{(9)} \oplus O_2^{(10)}$

f_i 関数の計算回数は 7 回である． □

上記の手順において，$\tilde{K}_1$ に関連する計算はステップ 4 以降の 5 回の f_i 関数の計算である．2^{52} 通りの $(I_8^{(5)}, \tilde{K}_1)$ を探索する際に，2^{20} 通りの $I_8^{(5)}$ を探索し，各 $I_8^{(5)}$ に対してまずステップ 1, 2, 3 を実行してから 2^{32} 通りの $\tilde{K}_1$ を探索するように実装すると，2^{26} 回の f_i 関数の計算は無視できるとすれば，5×2^{52} 回の f_i 関数の計算でアルゴリズムを実行できる．実際には次のような実装が可能である．

1. $I_1^{(1)} \in \mathcal{I}_1$, $I_7^{(6)} \in \mathcal{I}_7$ を満たす $I_1^{(1)}$, $I_7^{(6)}$ を任意に固定し，$O_1^{(1)}$, $O_7^{(6)}$ を計算する．
2. 2^{20} 通りの $I_8^{(5)}$ に対し，以下のステップ $3, 4, \ldots, 14$ を実行する．
3. $O_8^{(5)} \leftarrow f_8(I_8^{(5)})$
4. $O_2^{(2)} \leftarrow \tilde{C}_5 \oplus I_8^{(5)} \oplus O_8^{(5)}$
5. $\tilde{K}_2 \leftarrow f_2^{-1}(O_2^{(2)}) \oplus \tilde{C}_1$
6. 2^{32} 通りの $\tilde{K}_1$ に対し，以下のステップ $7, 8, \ldots, 14$ を実行する．
7. $O_3^{(3)} \leftarrow f_3(\tilde{K}_1 \oplus \tilde{C}_2 \oplus O_2^{(2)})$
8. $O_4^{(4)} \leftarrow \tilde{C}_4 \oplus I_8^{(5)} \oplus O_2^{(2)} \oplus O_3^{(3)}$
9. $O_6^{(7)} \leftarrow f_6(C_1 \oplus C_2 \oplus C_4 \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_7^{(6)})$
10. $O_5^{(8)} \leftarrow f_5(C_1 \oplus C_2 \oplus C_3 \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_3^{(3)} \oplus O_8^{(5)} \oplus O_7^{(6)} \oplus O_6^{(7)})$
11. $O_1^{(9)} \leftarrow f_1(I_1^{(1)} \oplus O_2^{(2)} \oplus O_3^{(3)} \oplus O_4^{(4)} \oplus O_7^{(6)} \oplus O_6^{(7)} \oplus O_5^{(8)})$
12. $O_2^{(10)} \leftarrow f_2(\tilde{K}_2 \oplus \tilde{C}_1 \oplus O_3^{(3)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_6^{(7)} \oplus O_5^{(8)} \oplus O_1^{(9)})$
13. $I_3^{(11)} \leftarrow \tilde{K}_1 \oplus C_1 \oplus C_3 \oplus I_1^{(1)} \oplus O_1^{(1)} \oplus O_2^{(2)} \oplus O_4^{(4)} \oplus O_8^{(5)} \oplus O_7^{(6)} \oplus O_5^{(8)} \oplus O_1^{(9)} \oplus O_2^{(10)}$
14. $I_3^{(11)} \in \mathcal{I}_3$ が成り立てば $(I_8^{(5)}, \tilde{K}_1)$ をファイルに保存する．

ファイルにはおおよそ 2^{26} 通りの $(I_8^{(5)}, \tilde{K}_1)$ が保存され，これらの $(I_8^{(5)}, \tilde{K}_1)$ は

$$I_1^{(1)} \in \mathcal{I}_1, I_7^{(6)} \in \mathcal{I}_7, I_3^{(11)} \in \mathcal{I}_3$$

を満たす．これらの $(I_8^{(5)}, \tilde{K}_1)$ に対し (K_1, K_2, K_3, K_4) を求め，$I_5^{(16)} \in \mathcal{I}_5$ が成り立つかをチェックする．成り立てば (K_1, K_2, K_3, K_4) を出力して終了する．

OK_1, OK_2 の導出には (C_1, C_2, C_3, C_4) を変えて 2 回アルゴリズムを実行する必要があり，計算量は合計で 10×2^{52} 回の f_i 関数の計算となる．256 ビット鍵 HyRAL の暗号化関数には 96 個の f_i 関数があるので，これは暗号化関数を $2^{48.8}$ 回実行する計算量に相当する．

8.7 計算機による等価鍵の導出

前章の等価鍵導出アルゴリズムを計算機上に実装した．計算機には名古屋大学情報基盤センターのスーパーコンピュータシステムを利用し，2 種類のサーバシステ

ム HX600 及び FX1 を用いた．HX600 は総ノード数 96 (384CPU/1536 コア)，総メモリ量 6TB，CPU は AMD Opteron 8380 (4 コア，2.5GHz) である．FX1 は総ノード数 768 (768CPU/3072 コア)，総メモリ量 24TB，CPU は SPARC64 VII (4 コア，2.52GHz) である．プログラムの実装には C 言語，プロセス並列処理のメッセージパッシングライブラリには MPI ライブラリを用いた．

δ, $I_1^{(1)}$, $I_7^{(6)}$ はそれぞれ $\delta = \texttt{0xd7d7d0d7}$, $I_1^{(1)} = \texttt{0x17170c17}$, $I_7^{(6)} = \texttt{0x1717292b}$ とした．$I_8^{(5)}$ の探索範囲によりプログラムを分割して実行し，各 $I_8^{(5)}$ に対し，$\tilde{K}_1$ は $\texttt{0x00000000}, \ldots, \texttt{0xffffffff}$ の 2^{32} 通りを全数探索した．$I_8^{(5)}$ の探索範囲とそれぞれのプログラムの計算時間を表 5 にまとめる．

OK_1 の導出には 2^{17} 通りの $I_8^{(5)}$ を，OK_2 の導出には 2^{20} 通りを探索した．$\tilde{K}_1$ と合わせて，OK_1 の導出には 2^{49} 通りの $(I_8^{(5)}, \tilde{K}_1)$ を探索し，OK_2 の導出には 2^{52} 通りを探索した．

実行の結果，表 4 にある 1 通りの OK_1 と 3 通りの OK_2 を導出することに成功した．

表 4: 等価鍵導出アルゴリズムの実行結果．

OK_1	0x2fd918837136d461f4bc99938907dd0b ($I_8^{(5)} = \texttt{0x00014b73}$, $\tilde{K}_1 = \texttt{0xdb658110}$)
OK_2	0xa20ed0f467141b2a3b038abb5f61d59e ($I_8^{(5)} = \texttt{0x0005b394}$, $\tilde{K}_1 = \texttt{0x990d5a4f}$)
	0xe3a1902aa60b6c3582a9131527d43b2f ($I_8^{(5)} = \texttt{0x000f8a7f}$, $\tilde{K}_1 = \texttt{0x6108833f}$)
	0x3218a5b25828a0b7d2122283894cc63b ($I_8^{(5)} = \texttt{0x000f9953}$, $\tilde{K}_1 = \texttt{0xe00a8731}$)

$\delta = \texttt{0xd7d7d0d7}$, $\Delta OK_1 = \Delta OK_2 = (\delta, \delta, \delta, \delta)$ と表 4 にある OK_1 と OK_2 に対し，

$$(K, K') = \begin{cases} ((OK_1, OK_2), (OK_1 \oplus \Delta OK_1, OK_2 \oplus \Delta OK_2)) \\ ((OK_1 \oplus \Delta OK_1, OK_2 \oplus \Delta OK_2), (OK_1, OK_2)) \\ ((OK_1 \oplus \Delta OK_1, OK_2), (OK_1, OK_2 \oplus \Delta OK_2)) \\ ((OK_1, OK_2 \oplus \Delta OK_2), (OK_1 \oplus \Delta OK_1, OK_2)) \end{cases} \tag{14}$$

の組み合わせはすべて等価鍵である．

表 5: OK_1, OK_2 の導出における $I_8^{(5)}$ の探索範囲と計算時間.

	システム名	キュー名	使用コア数	$I_8^{(5)}$ の探索範囲	$(I_8^{(5)}, \tilde{K}_1)$ の探索数	計算時間
OK_1	HX600	h1024	1024	0x00000000, . . . , 0x0000ffff	2^{48}	8 時間 48 分 56 秒
		h1024	1024	0x00010000, . . . , 0x0001ffff	2^{48}	8 時間 28 分 4 秒
OK_2	FX1	f1024	1024	0x00000000, . . . , 0x0003ffff	2^{50}	50 時間 36 分 2 秒
		f512	512	0x00040000, . . . , 0x0007ffff	2^{50}	92 時間 24 分 15 秒
	HX600	h256	256	0x00080000, . . . , 0x0009ffff	2^{49}	67 時間 42 分 47 秒
		h256	256	0x000a0000, . . . , 0x000bffff	2^{49}	67 時間 29 分 1 秒
		h256	256	0x000c0000, . . . , 0x000dffff	2^{49}	67 時間 34 分 55 秒
		h256	256	0x000e0000, . . . , 0x000fffff	2^{49}	67 時間 29 分 57 秒

以下，テストケースを示す．

$K = (OK_1, OK_2) \in \{0,1\}^{256}$ は秘密鍵，$M = (0, 0, \ldots, 0) \in \{0,1\}^{128}$ は平文，$C \in \{0,1\}^{128}$ は暗号文である．

- Case 1

$$\begin{cases} OK_1 & = \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 & = \texttt{0xa20ed0f467141b2a3b038abb5f61d59e} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0x1d470e813a1dafdca7cfd8d307ab4257} \end{cases}$$

- Case 2

$$\begin{cases} OK_1 & = \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 & = \texttt{0x75d90023b0c3cbfdecd45a6c88b60549} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0x1d470e813a1dafdca7cfd8d307ab4257} \end{cases}$$

- Case 3

$$\begin{cases} OK_1 & = \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 & = \texttt{0x75d90023b0c3cbfdecd45a6c88b60549} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0x0732ac003ba930da6955dc7bf79e9c5e} \end{cases}$$

- Case 4

$$\begin{cases} OK_1 & = \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 & = \texttt{0xa20ed0f467141b2a3b038abb5f61d59e} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0x0732ac003ba930da6955dc7bf79e9c5e} \end{cases}$$

- Case 5

$$\begin{cases} OK_1 & = \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 & = \texttt{0xe3a1902aa60b6c3582a9131527d43b2f} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0xb6fea87c1feba8ee3b30d67f5d710291} \end{cases}$$

- Case 6

$$\begin{cases} OK_1 & = \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 & = \texttt{0x347640fd71dcbce2557ec3c2f003ebf8} \\ M & = \texttt{0x00000000000000000000000000000000} \\ C & = \texttt{0xb6fea87c1feba8ee3b30d67f5d710291} \end{cases}$$

- Case 7

$$\begin{cases} OK_1 &= \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 &= \texttt{0x347640fd71dcbce2557ec3c2f003ebf8} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0x8eb2731da3cd6ec073cc85a2c8b46dde} \end{cases}$$

- Case 8

$$\begin{cases} OK_1 &= \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 &= \texttt{0xe3a1902aa60b6c3582a9131527d43b2f} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0x8eb2731da3cd6ec073cc85a2c8b46dde} \end{cases}$$

- Case 9

$$\begin{cases} OK_1 &= \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 &= \texttt{0x3218a5b25828a0b7d2122283894cc63b} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0x53403468cdd0ff20a623875f365d4df8} \end{cases}$$

- Case 10

$$\begin{cases} OK_1 &= \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 &= \texttt{0xe5cf75658fff706005c5f2545e9b16ec} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0x53403468cdd0ff20a623875f365d4df8} \end{cases}$$

- Case 11

$$\begin{cases} OK_1 &= \texttt{0x2fd918837136d461f4bc99938907dd0b} \\ OK_2 &= \texttt{0xe5cf75658fff706005c5f2545e9b16ec} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0xbbe6e08feadcec44b547e3c44cbe9a09} \end{cases}$$

- Case 12

$$\begin{cases} OK_1 &= \texttt{0xf80ec854a6e104b6236b49445ed00ddc} \\ OK_2 &= \texttt{0x3218a5b25828a0b7d2122283894cc63b} \\ M &= \texttt{0x00000000000000000000000000000000} \\ C &= \texttt{0xbbe6e08feadcec44b547e3c44cbe9a09} \end{cases}$$

8.8 本章のまとめ

本章ではまず，256 ビット鍵 HyRAL には等価鍵が $2^{51.0}$ 個 ($2^{50.0}$ ペア) 存在することを示した．さらに，計算量が $2^{48.8}$ である等価鍵導出アルゴリズムを設計した．また，設計したアルゴリズムを計算機上で実行しその正しさを実験的に検証するとともに，256 ビット鍵 HyRAL の等価鍵の具体例を導出できることを実証した．

一般に鍵長 k ビットのブロック暗号に対し，2^k の計算量で全数探索攻撃ができる．これを下回る攻撃の存在は，そのブロック暗号が理論的に解読されたことを意味する．$2^{50.0}$ ペアの等価鍵の存在は全数探索攻撃にかかる計算量が $2^{256} - 2^{50.0}$ であることを意味する．これは 2^{256} を下回っており，したがって 256 ビット鍵 HyRAL は理論的に解読された．

また，10 章で示すように，等価鍵の具体例は 256 ビット鍵 HyRAL を圧縮関数やハッシュ関数の構成に用いた場合の衝突の具体例が導出できたことを意味する．

9　弱鍵 (256 ビット鍵)

弱鍵とは，一般にほかの鍵よりも弱い暗号化関数になる鍵のことをいう [35]．DES に対しては，すべての M に対し

$$\mathrm{DES}_K(\mathrm{DES}_K(M)) = M$$

が成立する鍵 K が存在し，そのような鍵を弱鍵という [27]．また，すべての M に対し

$$\mathrm{DES}_{K_1}(\mathrm{DES}_{K_2}(M)) = M$$

が成立するような鍵のペア (K_1, K_2) を準弱鍵という [27]．

256 ビット鍵 HyRAL に対し，弱鍵 K を「すべての平文 M について $E_K(M) = E_{K'}(M)$ が成立するような $K' \neq K$ が存在する」という性質を持つ鍵と定義すると，等価鍵が $2^{51.0}$ 個存在することより，弱鍵は $2^{51.0}$ 個存在することになる．

10　ハッシュ関数での利用 (256 ビット鍵)

10.1　ハッシュ関数と衝突困難性

ブロック暗号は汎用的なセキュリティ技術であり，暗号化モード，メッセージ認証コード，認証暗号化モード，ハッシュ関数の構成などに用いられる．とくに HyRAL については，自己評価書 [16, 1 章] においてハッシュ関数への適用が設計思想として述べられている．ハッシュ関数 $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ は様々な用途に利用され，次の安全性要件を満たすことが要求される．

- 衝突困難性：$\mathcal{H}(M) = \mathcal{H}(M')$ となる互いに異なる (M, M') を計算することは困難でなければならない．(M, M') は $\mathcal{H}$ の衝突といわれる．
- 原像計算困難性：$Y \in \{0,1\}^n$ が与えられたとき $Y = \mathcal{H}(M)$ を満たす M を計算することは困難でなければならない．
- 第 2 原像計算困難性：$M \in \{0,1\}^*$ が与えられたとき $\mathcal{H}(M) = \mathcal{H}(M')$ を満たす M とは異なる M' を計算することは困難でなければならない．

本報告書では衝突困難性について考える．

10.2　256 ビット鍵 HyRAL による Davies-Meyer 方式圧縮関数の解析

ブロック長 n ビット，鍵長 k ビットのブロック暗号 E から圧縮関数 $h : \{0,1\}^{n+k} \to \{0,1\}^n$ を構成する方式として Davies-Meyer 方式が知られており [14, 6 章]，これは

$$h(H, M) = E_M(H) \oplus H$$

と定義される．図 14 にこれを示す．

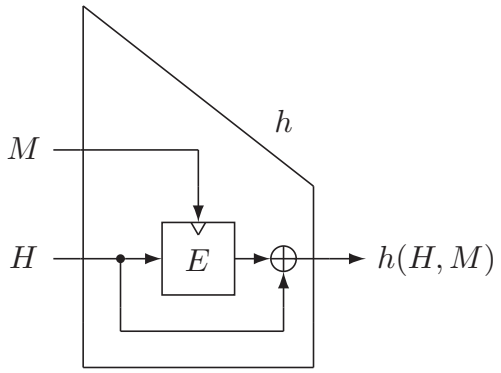

図 14: Davies-Meyer 方式．ブロック暗号の三角は鍵入力を表す．

E を 256 ビット鍵 HyRAL とし，(M, M') を式 (14) にある (K, K') のうちの一つとすると，任意の H に対して

$$h(H, M) = h(H, M')$$

が成立する．これは，256 ビット鍵 HyRAL から Davies-Meyer 方式により構成した圧縮関数に対し，2^{128} 通りの衝突 $((H, M), (H, M'))$ の具体例を導出できることを意味する．

10.3 256 ビット鍵 HyRAL を用いた Merkle-Damgård 方式によるハッシュ関数の解析

次に，圧縮関数 $h : \{0,1\}^{n+k} \to \{0,1\}^n$ からハッシュ関数 $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ を構成する一般的な方式として Merkle-Damgård 方式が知られている [14, 6 章]．

$IV \in \{0,1\}^n$ を初期値とし，入力ビット列 $M \in \{0,1\}^*$ を適切にパディングしたビット列 $\tilde{M} = (M_1, M_2, \ldots, M_m) \in \{0,1\}^{mn}$ に対し，出力 $\mathcal{H}(M)$ は次のように計算される．まず

$$H_0 \leftarrow IV$$

とし，次に $i = 1, 2, \ldots, m$ に対し

$$H_i \leftarrow h(H_{i-1}, M_i)$$

とする．最後に $\mathcal{H}(M) \leftarrow H_m$ を出力する．図 15 にこれを示す．

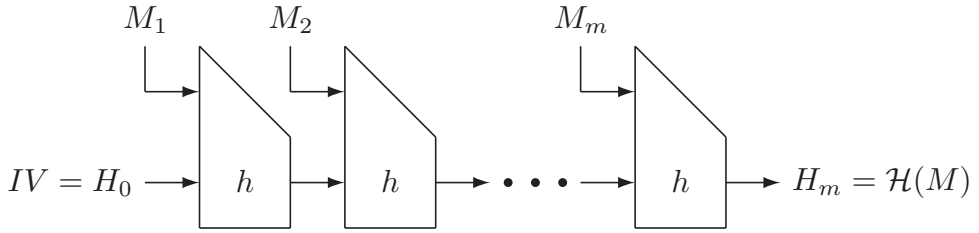

図 15: Merkle-Damgård 方式．

E を 256 ビット鍵 HyRAL とし，h を E から Davies-Meyer 方式により構成された圧縮関数とし，$\mathcal{H}$ を h から Merkle-Damgård 方式により得られたハッシュ関数とする．

このとき，式 (14) にある任意の (K, K') に対して，M, M' を

$$M, M' \in \{K, K'\}^m, M \neq M'$$

を満たすようにすると，SHA-1 などで用いられている一般的な 0 と長さを連結するようなパディングの場合，$\mathcal{H}(M) = \mathcal{H}(M')$ が成り立ち，$\mathcal{H}$ に対する衝突が得られる．例えば $m = 3$ の場合，以下の例が考えられる．

$$(M, M') = \begin{cases} ((K, K, K), (K', K', K')) \\ ((K, K, K'), (K', K', K)) \\ ((K, K', K), (K', K, K')) \\ ((K, K', K'), (K', K, K)) \end{cases}$$

同様にして，m ブロックの M と M' を考えた場合，2^{m-1} 通りの衝突が得られる．

10.4 本章のまとめ

256 ビット鍵 HyRAL から Davies-Meyer 方式により構成した圧縮関数に対し，2^{128} 通りの衝突の具体例を導出できることを示した．また，この圧縮関数を用いて Merkle-Damgård 方式でハッシュ関数を構成した場合，上記の圧縮関数に対する衝突を用いることによりハッシュ関数に対する衝突も容易に導出できることを示した．

11　関連鍵攻撃 (256 ビット鍵)

11.1　関連鍵攻撃の概要

Biham と Knudsen による関連鍵攻撃はブロック暗号に対する汎用的な攻撃法であり [9, 10, 21]，Bellare, Kohno により定式化され [5]，また Albrecht らにより理論的解析がされている [1]．本章では [5] の安全性定義に従い，256 ビット鍵 HyRAL の関連鍵攻撃に対する安全性を考える．

鍵長 k ビット，ブロック長 n ビットのブロック暗号 $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ を考える．また，鍵長 k ビット，ブロック長 n ビットの理想暗号 $G : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ を考える．これは，各 $K \in \{0,1\}^k$ に対し，$G_K(\cdot)$ が $\{0,1\}^n$ 上のランダム置換である暗号である．

敵はオラクルに対し，クエリとして (ϕ, M) を送信する．$\phi : \{0,1\}^k \to \{0,1\}^k$ は関連鍵導出関数と呼ばれ，$\phi \in \Phi$ である必要がある．敵がクエリに使用可能な関連鍵導出関数の集合 Φ は allowed 関連鍵導出関数と呼ばれる．クエリ (ϕ, M) に対し，オラクルがブロック暗号の場合は $E_{\phi(K)}(M)$ が返される．このようなオラクルを $E_{\mathrm{RK}(\cdot,K)}(\cdot)$ と表記する．オラクルが理想暗号の場合は $G_{\phi(K)}(M)$ が返される．このようなオラクルを $G_{\mathrm{RK}(\cdot,K)}(\cdot)$ と表記する．オラクルへアクセスしたのち，敵は 1 ビットを出力する．このときの敵 A の識別成功確率は

$$\mathbf{Adv}^{\text{prp-rka}}_{E,\Phi}(A) = \Pr[A^{E_{\mathrm{RK}(\cdot,K)}(\cdot)} \Rightarrow 1] - \Pr[A^{G_{\mathrm{RK}(\cdot,K)}(\cdot)} \Rightarrow 1]$$

と定義される．

11.2　256 ビット鍵 HyRAL に対する関連鍵攻撃

256 ビット鍵 HyRAL には等価鍵が存在するため，このことを用いて定数回のクエリにより関連鍵識別攻撃を行うことができる．

$\delta = \texttt{0xd7d7d0d7}$, $\Delta OK_1 = \Delta OK_2 = (\delta, \delta, \delta, \delta)$, $\Delta K = (\Delta OK_1, \Delta OK_2)$ とし，上記の一般的な関連鍵攻撃の特殊な場合である次の攻撃を考える．敵 A は関連鍵暗号化オラクルのペア $(E_K(\cdot), E_{K\oplus\Delta K}(\cdot))$ と 2 つの独立なランダム置換オラクルのペア $(P_1(\cdot), P_2(\cdot))$ を識別する．A の識別成功確率を

$$\mathbf{Adv}^{\text{prp-rka}}_{E,\Delta K}(A) = \Pr[A^{E_K(\cdot),E_{K\oplus\Delta K}(\cdot)} \Rightarrow 1] - \Pr[A^{P_1(\cdot),P_2(\cdot)} \Rightarrow 1]$$

と定義する．

このとき，ランダムに選ばれた K に対し，$(K, K \oplus \Delta K)$ は確率 2^{-206} で等価鍵となり，したがって $E_K(\cdot)$ と $E_{K\oplus\Delta K}(\cdot)$ は同じ入力に対し同じ出力を返す．敵のオラクルを $(O_1(\cdot), O_2(\cdot))$ と表記し，以下の敵を考える．

Step 1. q を適当な整数とし (例えば $q = 5$)，$M_1, \ldots, M_q$ を任意に固定する．ただし $M_i \neq M_j$ であり，各 M_i は 128 ビットである．

Step 2. $1 \leq i \leq q$ に対し，$C_i \leftarrow O_1(M_i)$ を計算する．

Step 3. $1 \leq i \leq q$ に対し，$D_i \leftarrow O_2(M_i)$ を計算する．

Step 4. すべての $1 \leq i \leq q$ に対し $C_i = D_i$ であれば 1 を，そうでなければ 0 を出力する．

$(O_1(\cdot), O_2(\cdot)) = (E_K(\cdot), E_{K \oplus \Delta K}(\cdot))$ のとき，敵が 1 を出力する確率は 2^{-206} 以上である．一方，$(O_1(\cdot), O_2(\cdot)) = (P_1(\cdot), P_2(\cdot))$ のとき，敵が 1 を出力する確率は $(2^{128} - q)!/(2^{128})!$ である．したがって

$$\mathbf{Adv}^{\text{prp-rka}}_{E,\Delta K}(A) \geq \frac{1}{2^{206}} - \frac{(2^{128} - q)!}{(2^{128})!}$$

となり，$q = 5$ の場合，確率 $2^{-206} - (2^{128} - 5)!/(2^{128})! \approx 2^{-206}$ で識別に成功する．

12　関連暗号攻撃 $(129, 130, \ldots, 256$ ビット鍵$)$

12.1　関連暗号攻撃の概要

本章では Wu により提案された関連暗号攻撃を考える [36]．一般に秘密鍵は一つの暗号に対して使用されるが，この攻撃ではいくつかの異なる暗号が同じ秘密鍵を使用している状況を考える．一般に，同じ秘密鍵が異なるブロック暗号で使用されたとしてもそれが直ちに脆弱性につながることはない．例えば AES に対しては，AES-128 の鍵を繰り返した鍵を AES-256 で使用したとしてもそれが直ちに脆弱性につながることはない [36]．

12.2　$129, 130, \ldots, 256$ ビット鍵 HyRAL に対する関連暗号攻撃

HyRAL の暗号化アルゴリズムは鍵長が 128 ビットの場合と $129, 130, \ldots, 256$ ビットの場合で構造が異なる．鍵長が 256 ビットの場合，秘密鍵 K の上位 128 ビットを OK_1，下位 128 ビットを OK_2 とする．鍵長が k ビット $(129 \leq k \leq 255)$ の場合，K の上位 128 ビットを OK_1 とし，下位 $k-128$ ビットに $256-k$ ビット分の 0 を連結したビット列を OK_2 とする．

ここで，$K^{(256)}$ を 256 ビット鍵 HyRAL ($E^{(256)}$ と表記する) の秘密鍵とする．また，$K^{(k)}$ を $K^{(256)}$ の上位 k ビットとし，k ビット鍵 HyRAL ($E^{(k)}$ と表記する) の秘密鍵とする．

仮に敵が $E^{(256)}_{K^{(256)}}(\cdot)$ オラクルと $E^{(255)}_{K^{(255)}}(\cdot)$ オラクルにアクセスできる場合，敵は任意の M に対する $E^{(256)}_{K^{(256)}}(M)$ と $E^{(255)}_{K^{(255)}}(M)$ を比較することにより，$K^{(256)}$ の最下位 1 ビットを特定することができる．

$K^{(255)}$ より得られる OK_2 の最下位ビット ($OK_2^{(255)}$ と表記する) は必ず 0 であり，$K^{(256)}$ から得られる OK_2 の最下位ビット ($OK_2^{(256)}$ と表記する) は 0 の場合と 1 の場合がある．仮に $OK_2^{(256)}$ の最下位ビットが 0 の場合，$OK_2^{(256)} = OK_2^{(255)}$ であるから任意の M に対して $E^{(256)}_{K^{(256)}}(M) = E^{(255)}_{K^{(255)}}(M)$ が成立する．一方，$OK_2^{(256)}$ の最下位ビットが 1 の場合，$OK_2^{(256)} \neq OK_2^{(255)}$ であるから $E^{(256)}_{K^{(256)}}(M) = E^{(255)}_{K^{(255)}}(M)$ が成立する確率は低い．

同様に，$E^{(129)}_{K^{(129)}}(\cdot), E^{(130)}_{K^{(130)}}(\cdot), \ldots, E^{(256)}_{K^{(256)}}(\cdot)$ オラクルにアクセスできる場合，敵は $K^{(256)}$ の下位 127 ビットを以下のように導出できる．以下，$K_i^{(256)}$ は $K^{(256)}$ の i 番目のビット $(1 \leq i \leq 256)$ を表す．

Step 1. 128 ビットの M を任意に固定する．

Step 2. $E^{(256)}_{K^{(256)}}(M) = E^{(255)}_{K^{(255)}}(M)$ であれば $K_{256}^{(256)} = 0$ であり，そうでなければ $K_{256}^{(256)} = 1$ である．

Step 3. $E^{(255)}_{K^{(255)}}(M) = E^{(254)}_{K^{(254)}}(M)$ であれば $K^{(256)}_{255} = 0$ であり，そうでなければ $K^{(256)}_{255} = 1$ である．

$\vdots$

Step 127. $E^{(131)}_{K^{(131)}}(M) = E^{(130)}_{K^{(130)}}(M)$ であれば $K^{(256)}_{131} = 0$ であり，そうでなければ $K^{(256)}_{131} = 1$ である．

Step 128. $E^{(130)}_{K^{(130)}}(M) = E^{(129)}_{K^{(129)}}(M)$ であれば $K^{(256)}_{130} = 0$ であり，そうでなければ $K^{(256)}_{130} = 1$ である．

上記の攻撃により，敵は $K^{(256)}_{130}K^{(256)}_{131}, \ldots, K^{(256)}_{256}$ の 127 ビットを特定できる．またこの攻撃に必要なクエリ回数は 128 回である．

13　圧縮関数での利用 ($129, 130, \ldots, 256$ ビット鍵)

$E^{(256)}$ から Davies-Meyer 方式により得られた圧縮関数を $h^{(256)}$ とし，$E^{(192)}$ から Davies-Meyer 方式により得られた圧縮関数を $h^{(192)}$ とする．$h^{(256)}$ と $h^{(192)}$ は異なるブロック暗号から構成される圧縮関数であり，これらには何ら関連性がないことが要求される．

ここで，H を任意の 128 ビットのビット列，M' を任意の 192 ビットのビット列とし，M' に 64 ビット分の 0 を連結したビット列を M とすると

$$h^{(256)}(H, M) = h^{(192)}(H, M')$$

が成立し，出力が衝突を起こす．

同様に，$E^{(129)}, E^{(130)}, \ldots, E^{(256)}$ から得られる圧縮関数を $h^{(129)}, h^{(130)}, \ldots, h^{(256)}$ とするとき，異なる圧縮関数 $h^{(k)}$ と $h^{(k')}$ に対し

$$h^{(k)}(H, M) = h^{(k')}(H, M')$$

を満たす (H, M) と (H, M') を構成できる．

14　その他の攻撃

これまでに指摘した以外に HyRAL に対する問題点は発見していない.

15 まとめ

評価期間中に得られた結果をまとめる.

- 256 ビット鍵 HyRAL には等価鍵が $2^{51.0}$ 個 ($2^{50.0}$ ペア) 存在することを示し, 256 ビット鍵 HyRAL を理論的に解読した.
- 256 ビット鍵 HyRAL の等価鍵の具体例を計算機により導出した.
- 256 ビット鍵 HyRAL に対し, $2^{51.0}$ 個の弱鍵が存在することを示した.
- 256 ビット鍵 HyRAL を用いて Davies-Meyer 方式により構成した圧縮関数に対し, 衝突の具体例を導出できることを示した.
- 上記の圧縮関数を用いて Merkle-Damgård 方式でハッシュ関数を構成した場合, このハッシュ関数に対する衝突の具体例を導出できることを示した.
- 256 ビット鍵 HyRAL に対し, 識別成功確率が 2^{-206} である関連鍵識別攻撃が可能であることを示した.
- $129, 130, \ldots, 256$ ビット鍵 HyRAL に対する関連暗号攻撃により, 128 回のクエリで 256 ビット鍵 HyRAL の秘密鍵の下位 127 ビットを導出できることを示した.
- $129, 130, \ldots, 256$ ビット鍵 HyRAL を用いて Davies-Meyer 方式により構成した圧縮関数に対し, 異なるブロック暗号を用いた圧縮関数同士の衝突を導出できることを示した.
- その他の問題点は発見していない.

謝辞

本報告書は名古屋大学工学部電気電子・情報工学科，浅野優貴君と名古屋大学大学院工学研究科計算理工学専攻，柳原慎吾君の協力を得て作成した．計算機実験には名古屋大学情報基盤センターのスーパーコンピュータシステムを利用した．計算機実験に関して助言をいただいた名古屋大学大学院工学研究科安藤秀樹教授に感謝する．

参考文献

[1] M.R. Albrecht, P. Farshim, K.G. Paterson, G.J. Watson, "On Cipher-Dependent Related-Key Attacks in the Ideal-Cipher Model," Pre-proceedings of FSE 2011.

[2] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms," Available at `http://info.isl.ntt.co.jp/crypt/camellia/dl/support.pdf`, 2000.

[3] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms," SAC 2000, LNCS 2012, pp. 41–54, Springer-Verlag, 2001.

[4] 浅野優貴, 柳原慎吾, 岩田哲, "256 ビット鍵 HyRAL の等価鍵," SCIS 2011, 2B2-3, 2011.

[5] M. Bellare, T. Kohno, "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications," EUROCRYPT 2003, LNCS 2656, pp. 491–506, Springer-Verlag, 2003.

[6] E. Biham, A. Biryukov, A. Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," EUROCRYPT '99, LNCS 1592, pp. 12–23, Springer-Verlag, 1999.

[7] E. Biham, A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard," Springer-Verlag, 1993.

[8] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," J. Cryptology, 4 (1), pp. 3–72, Springer-Verlag, 1991.

[9] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys (Extended Abstract)," EUROCRYPT '93, LNCS 765, pp. 398–409, Springer-Verlag, 1993.

[10] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," J. Cryptology, 7 (4), pp. 229–246, Springer-Verlag, 1994.

[11] N.T. Courtois, A. Klimov, J. Patarin, A. Shamir, "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations," EUROCRYPT 2000, LNCS 1807, pp. 392–407, Springer-Verlag, 2000.

[12] N.T. Courtois, J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined System of Equations," ASIACRYPT 2002, LNCS 2501, pp. 267–287, Springer-Verlag, 2002.

[13] Cryptography Research and Evaluation Committees (CRYPTREC), http://www.cryptrec.go.jp/index.html

[14] 電子情報通信学会, “情報セキュリティハンドブック,” オーム社, 2004.

[15] 平田耕蔵, “共通鍵 128 ビットブロック暗号 HyRAL,” SCIS 2010, 1D1-1, 2010.

[16] 平田耕蔵, “HyRAL 自己評価書,” CRYPTREC 応募書類, 2010.

[17] Kouzou Hirata, “The 128 Bit Block Cipher HyRAL (Hybrid Randomization Algorithm) Common Key Block Cipher,” IPTC 2010, 2010.

[18] T. Jakobsen, L.R. Knudsen, “The Interpolation Attack on Block Ciphers,” FSE ’97, LNCS 1267, pp. 28–40, Springer-Verlag, 1997.

[19] 通信・放送機構, “共通鍵ブロック暗号の選択/設計/評価に関するドキュメント,” 情報セキュリティ技術に関する研究開発プロジェクト, 2000.

[20] L.R. Knudsen, “Cryptanalysis of LOKI,” ASIACRYPT ’91, LNCS 739, pp. 22–35, Springer-Verlag, 1993.

[21] L.R. Knudsen, “Cryptanalysis of LOKI91,” AUSCRYPT ’92, LNCS 718, pp. 196–208, Springer-Verlag, 1992.

[22] L.R. Knudsen, “Truncated and Higher Order Differentials,” FSE ’94, LNCS 1008, pp. 196–211, Springer-Verlag, 1995.

[23] 五十嵐保隆, 高木幸弥, 金子敏信, “共通鍵ブロック暗号 HyRAL の線形攻撃耐性評価,” SCIS 2010, 1D1-3, 2010.

[24] X. Lai, “Higher Order Derivatives and Differential Cryptanalysis,” Proceedings of Symposium on Communication, Coding and Cryptography, in honor of J.L. Massey on the occasion of his 60th birthday, 1994.

[25] M. Matsui, “Linear Cryptanalysis Method for DES Cipher,” EUROCRYPT ’93, LNCS 765, pp. 386–397, Springer-Verlag, 1994.

[26] M. Matsui, “The First Experimental Cryptanalysis of the Data Encryption Standard,” CRYPTO ’94, LNCS 839, pp. 1–11, Springer-Verlag, 1994.

[27] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, “Handbook of Applied Cryptography,” CRC Press, 1996.

[28] 角尾幸保, 辻原悦子, 中嶋浩貴, 久保博靖, “変形 Feistel 構造を持つブロック暗号の不可能差分,” SCIS 2007, 4A2-2, 2007.

[29] 芝山直喜, 五十嵐保隆, 金子敏信, 半谷精一郎, “共通鍵ブロック暗号 HyRAL の不能差分攻撃について,” FIT2010, L-022, 2010.

[30] 芝山直喜, 五十嵐保隆, 金子敏信, 半谷精一郎, “共通鍵ブロック暗号 HyRAL の MDS 行列の分岐数を利用した不能差分特性について,” 2010 年電子情報通信学会基礎・境界ソサイエティ大会, A-7-8, 2010.

[31] 芝山直喜, 五十嵐保隆, 金子敏信, 半谷精一郎, “共通鍵ブロック暗号 HyRAL の飽和攻撃耐性評価,” SITA 2010, 10.1, 2010.

[32] 芝山直喜, 五十嵐保隆, 金子敏信, 半谷精一郎, “共通鍵ブロック暗号 HyRAL に対する高階差分攻撃,” ISEC 2011 (2011-03), 2011.

[33] 多賀文吾, 田中秀磨, “共通鍵ブロック暗号 HyRAL の高階差分特性,” SCIS 2011, 2B2-2, 2011.

[34] 高木幸弥, 五十嵐保隆, 金子敏信, “共通鍵ブロック暗号 HyRAL の差分攻撃耐性評価,” SCIS 2010, 1D1-2, 2010.

[35] Henk C.A. van Tilborg, “Encyclopedia of Cryptography and Security,” Springer, 2005.

[36] H. Wu, “Related-Cipher Attacks,” ICICS 2002, LNCS 2513, pp. 447–455, Springer-Verlag, 2002.

[37] 山口洋平, 五十嵐保隆, 金子敏信, “共通鍵ブロック暗号 HyRAL の高階差分特性,” 第 63 回電気関係学会九州支部連合大会, 02-1A-06, 2010.